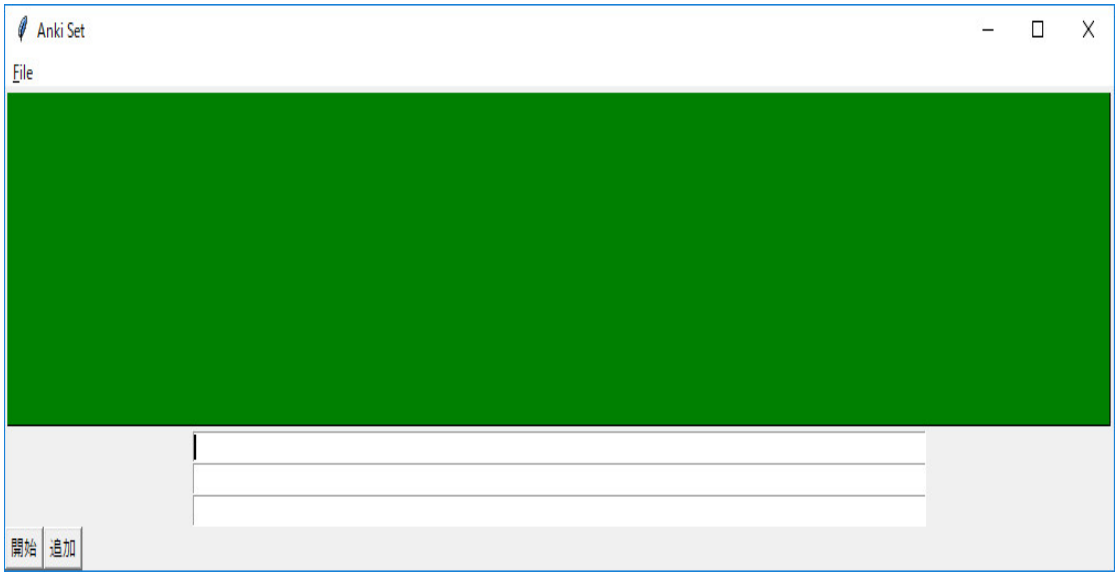
高知大学教育学部の情報数学のテキスト：おまけ

文責　：　高知大学名誉教授　中村　治

＜単語カードの為のプログラム＞

インターネットで探すと英単語を覚えるための高機能（例えば、記憶のメカニズムを利用したとか）をうたったソフトが幾つかありますが、私にとっては使いにくいです。私は外国語の勉強をするのが好きで、英語、ドイツ語、ロシア語、フランス語、アラビア語、… と文法を学んできました（尤も勉強の仕方が分からず、ものになった外国語は一つもないですが）が、外国語の単語を覚えるのが苦手で、今まで専門用語の学習と文脈から意味の類推で、何とかしのいできました。定年退職して、暇になったので、外国語の単語を本気になって覚えてみようと思って、インターネットで、色々な人の外国語の単語の覚え方を探索しましたが、述べられているどの方法も私には続けることが出来ませんでした。しかし、英単語を少しは覚えています。そこで、中学生の時、どうやって英単語を覚えたか思い出してみると、手のひらサイズの機械で、英単語とその訳を印刷した厚紙を、シャカシャカと次々に表示させて覚えたことを思い出しました。機械仕掛けのフラッシュカードです。私にとっては、この方法が唯一の英単語を覚えることが出来た方法みたいです。中学生時代の機械を自作することは私には出来ませんので、ソフトで作ってみましょう。言語は何でも良いですが、ここでは単純な単語カードを表示するソフトを **Python 3.6** で作ってみましょう。

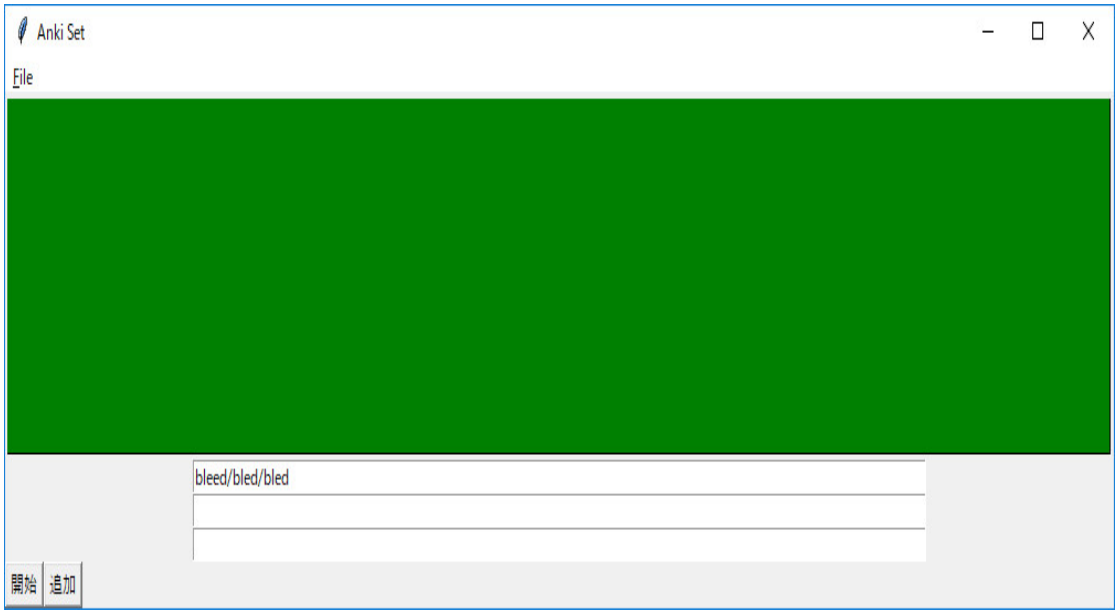
ここで作るプログラムは２つです。まずはカードのデータを作るソフトです。立ち上げると



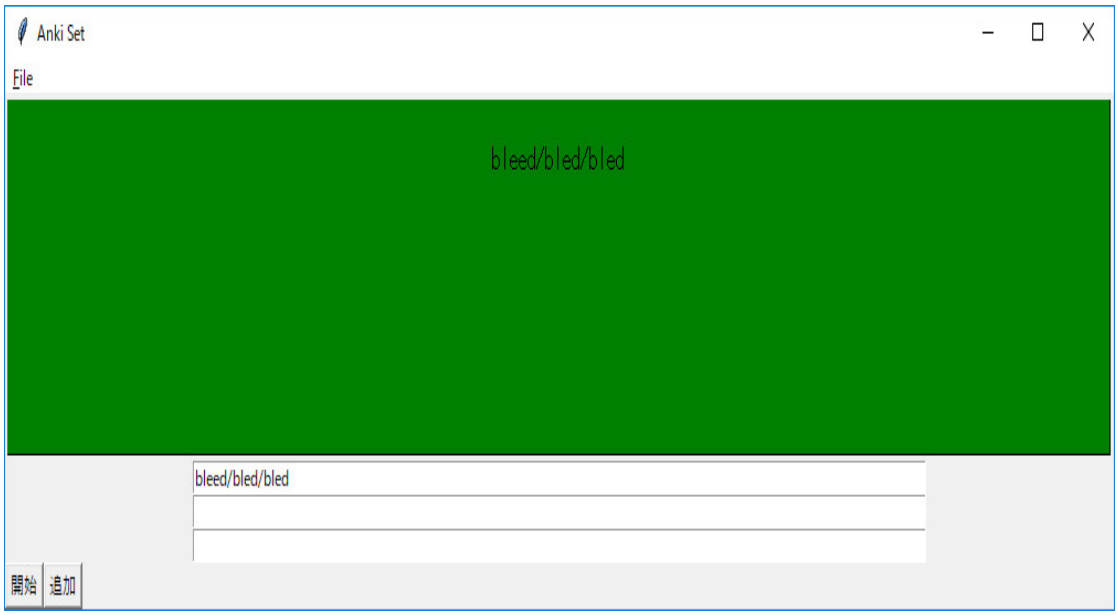
です。表示されている図は縦横の比率が実際と異なっています。各自自分のパソコンに合わせて、見やすい大きさにサイズを修正すれば良いです。修正の方法は簡単です。「File」メニューには「保存」と「読込」があります。「保存」は作成したデータをファイルに保存します。「読込」は作りかけのデータファイルにデータを追加するために、作りかけのデータファイルを読み込み、データを追加します。

データを作成するには、「開始」のボタンをクリックします。「開始」のボタンをクリックすると辞書が初期化されます。従って、「読込」で作りかけのデータファイルを読み込んだ時には、辞書に作りかけのファイルのデータが既にセットされているので、「開始」のボタンをクリックすると辞書が初期化され、データの追加が出来ないので注意してください。

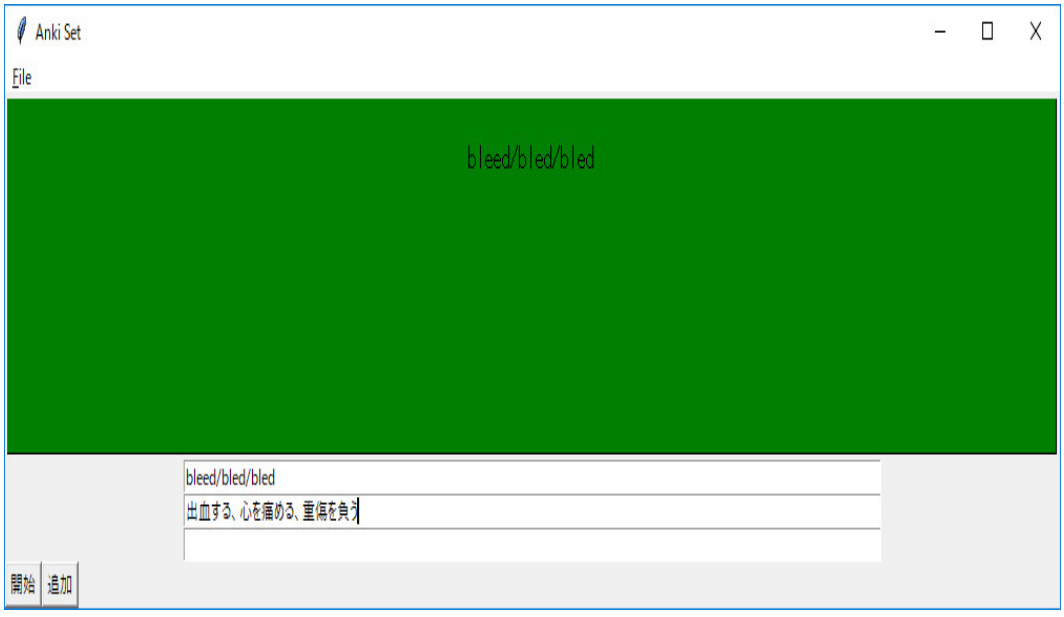
辞書にデータを追加するには、まず上のエントリー（テキストエディタ）に英単語（日本語でもいい）を入力します。例えば、bleed/bled/bled と入力します。



エンターキーを押します。



bleed/bled/bled が表示され、辞書のキーがセットされました。次に、中央のエントリー（テキストエディタ）に日本語（英単語でもいい）を入力します。ここでは、bleed の訳：「出血する、心を痛める、重傷を負う」を入力します。



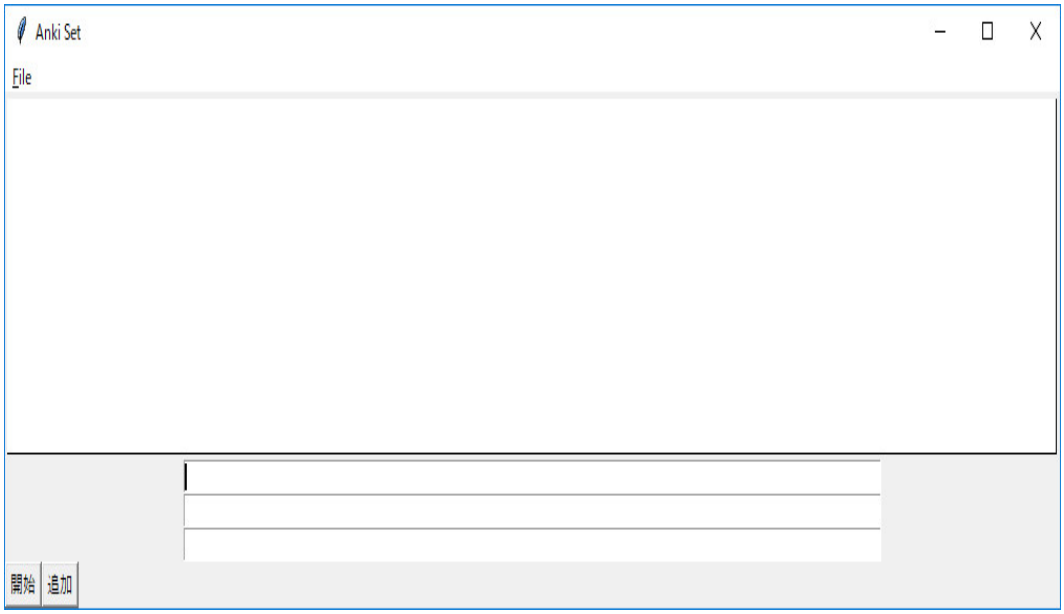
エンターキーを押します。

「出血する、心を痛める、重傷を負う」が表示され、辞書の値がセットされました。次に、舌のエントリー（テキストエディタ）に bleed に関する j 付加的な情報を入力します。ここでは、「The driver was bleeding heavily when the ambulance arrived at the accident site.」と入力します。

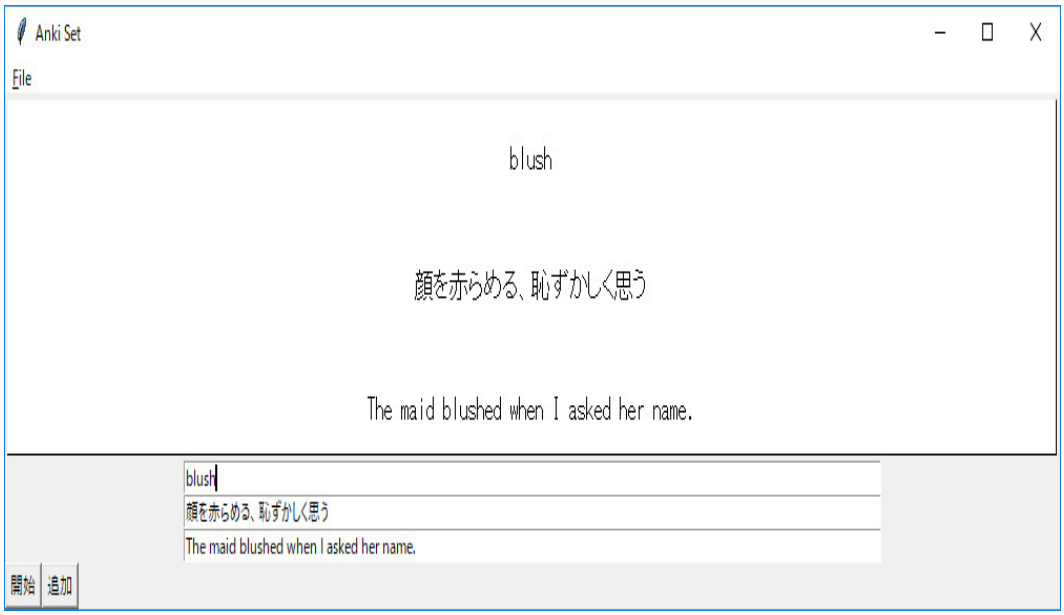
エンターキーを押します。

「The driver was bleeding heavily when the ambulance arrived at the accident site.」が表示され、短文の値がセットされました。

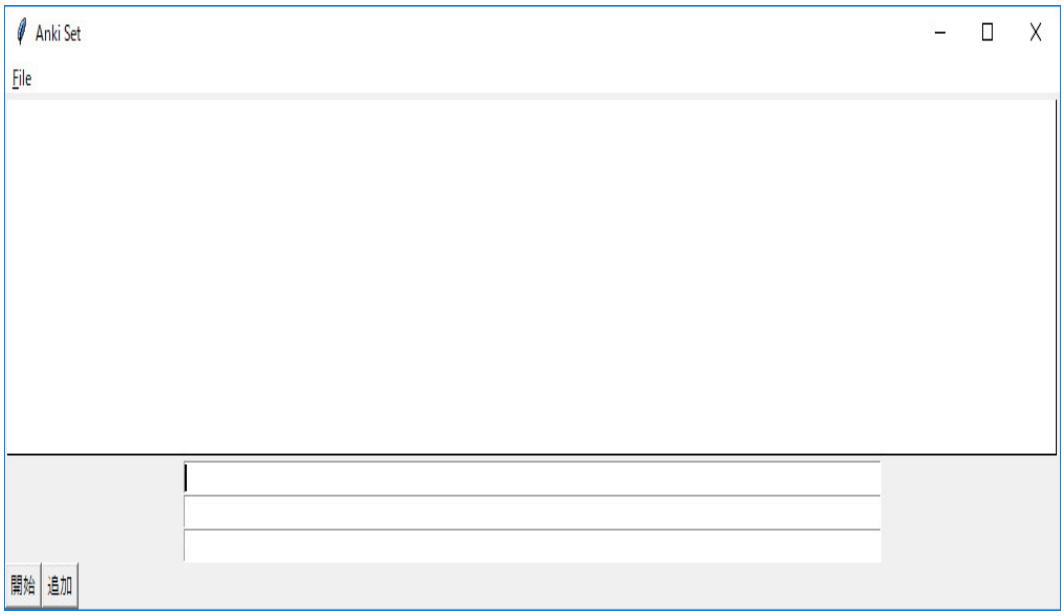
　これで、辞書のキーと値と短文が揃ったので、実際に辞書にそのデータを追加するために、「追加」のボタンをクリックします。



次のデータを上のエントリー（テキストエディタ）と下のエントリー（テキストエディタ）に入力します。



「追加」のボタンをクリックします。



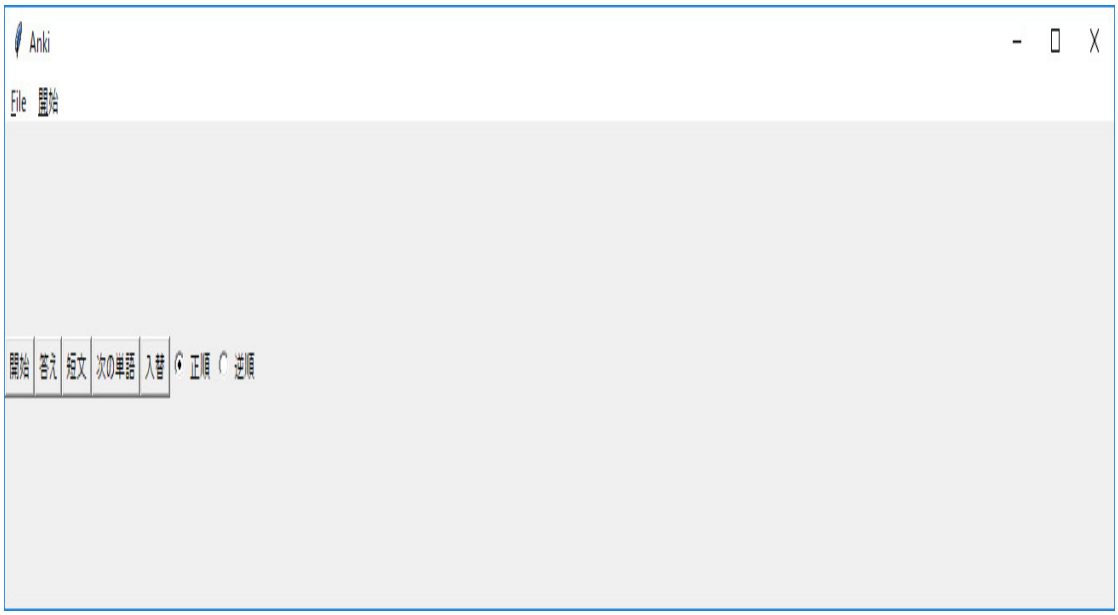
これを繰り返して、覚えたい単語のデータを作ります。データが出来たら、ファイルに保存します。「File」メニューの「保存」を使って保存します。

保存したファイルは


```
[
(bleed/bled/bled:出血する、心を痛める、重傷を負う:
The driver was bleeding heavily when the ambulance arrived
at the accident site.)
(blush:顔を赤らめる、恥ずかしく思う:The maid blushed when I asked her name.)
]
```

のようなテキストファイルです。（実際は上の画面のように 4 行です。）最初の行は [だけ、最後の行は] だけ、辞書の中身は (キーの単語:訳:短文 です。[ ] (:) はすべて半角です。[ ] はデータには使ってはいけません。従って、全く同じ形式でデータを作ることによって、テキストエディターで、単語カードを編集することが出来ますが、文字コードの問題があって注意が必要です。後で解説します。

　次は、ここで作成したファイルを読み込んで、順番に表示するソフトです。立ち上げると



です。表示されている図は縦横の比率が異なっています。自分のパソコンに合わせて、プログラムを修正すれば良いです。「File」メニューには「読込」があります。「読込」で保存してあるファイルを読み込みます。先ほどのファイルを読み込んでみます。

「開始」のボタンかメニューの開始をクリックします。

最初の単語が表示されます。「答え」のボタンを押すと

キーと結び付けられている訳が表示されます。

「短文」のボタンを押すと

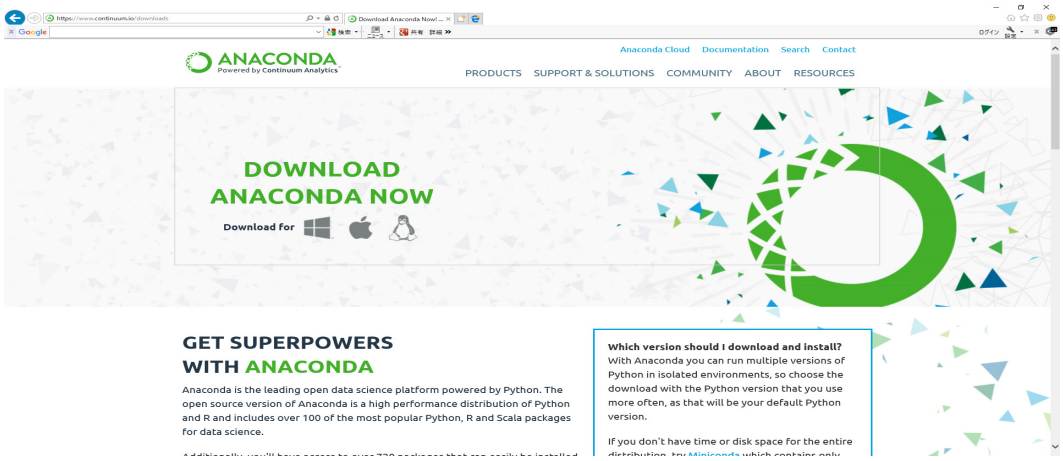
「次の単語」のボタンを押すと

次のキーが表示されます。「入替」のボタンを押すと辞書がランダムにシャッフルされます。「正順」と「逆順」のラジオボタンは、単語と訳の表示の順番を入れ替えるためのものです。

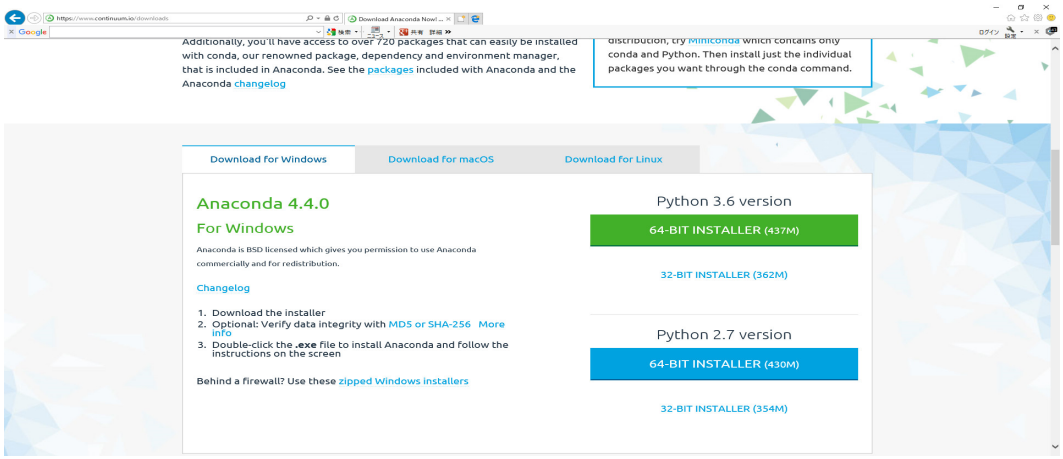
それではこのようなプログラムを作っていきましょう。まず Python をインストールして、Python を使えるようにしましょう。ここでは Anaconda を使って Python をインストールします。Anaconda は Python 本体に加えて、よく使われるパッケージを一括してインストールできるようにしたものです。

`https://www.continuum.io/downloads`

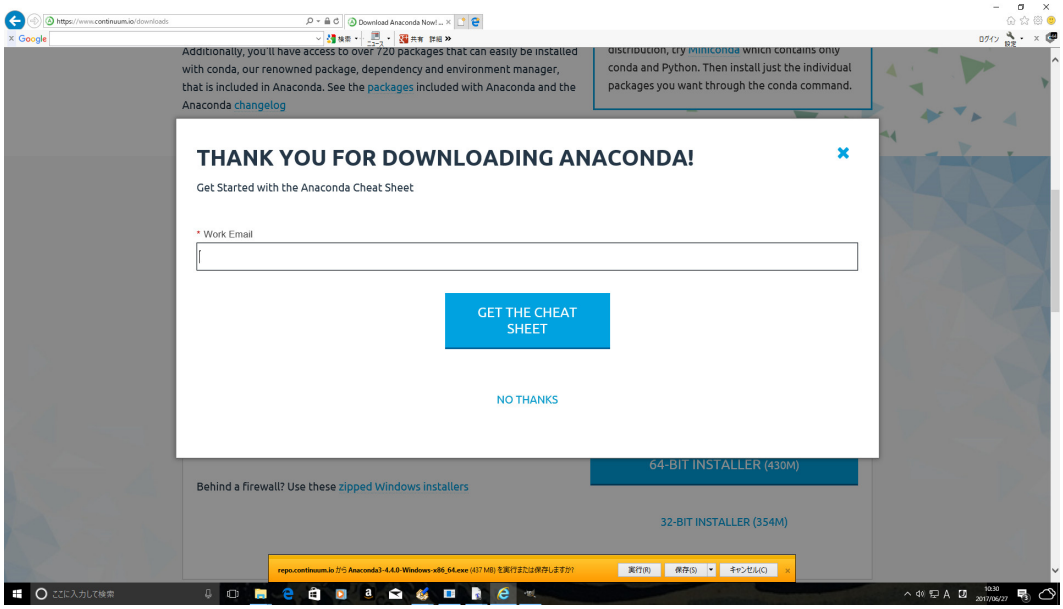
にアクセスします。



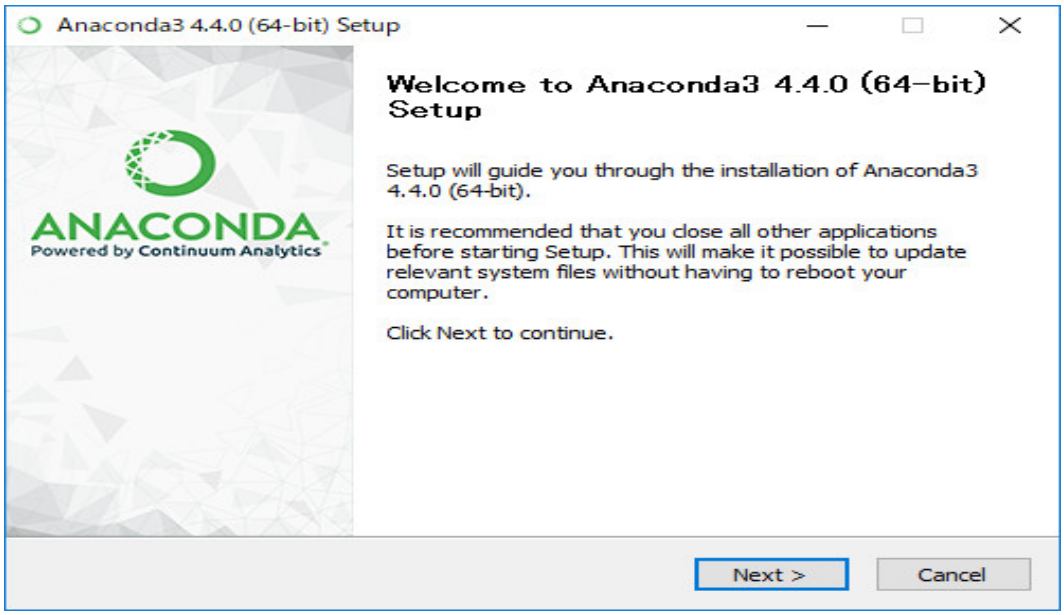
下の方の



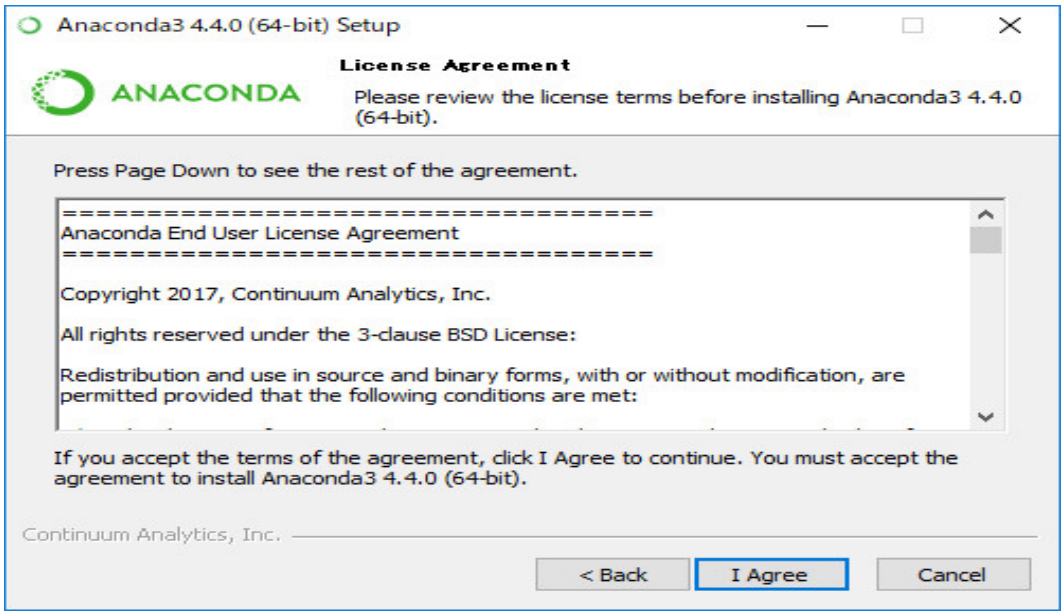
Python 3.6 version をクリックします。現在 Python2 と Python3 が利用できますが、Python 2.7 と Python 3.6 は上位互換性はなく、Python2 は新たな改善はなされないことが決まっているので、これからは Python3 を使って行くのが良いです。



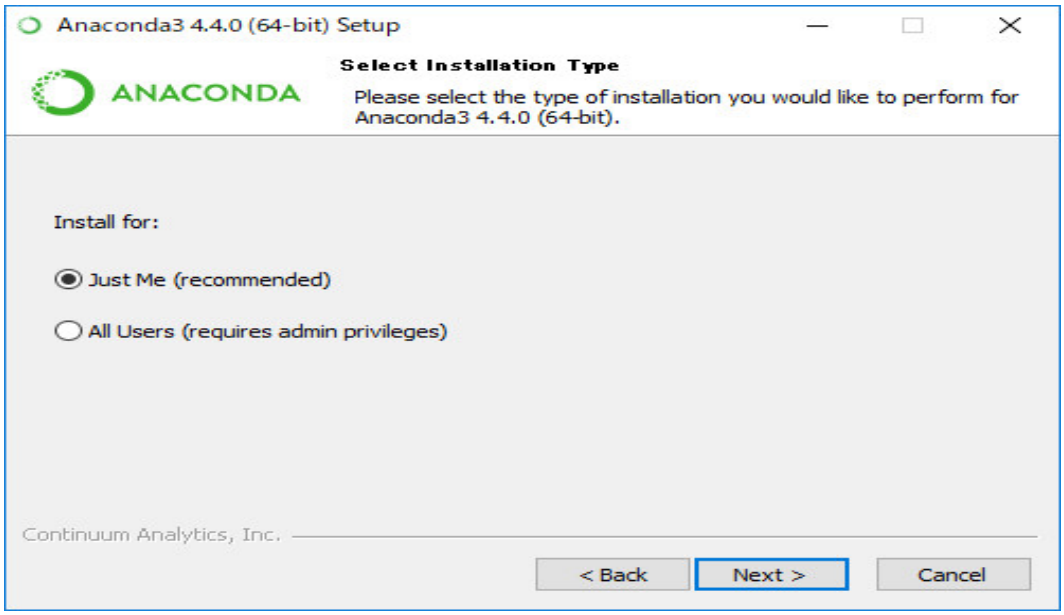
これは無視して、閉じて良いです。「実行」をクリックします。



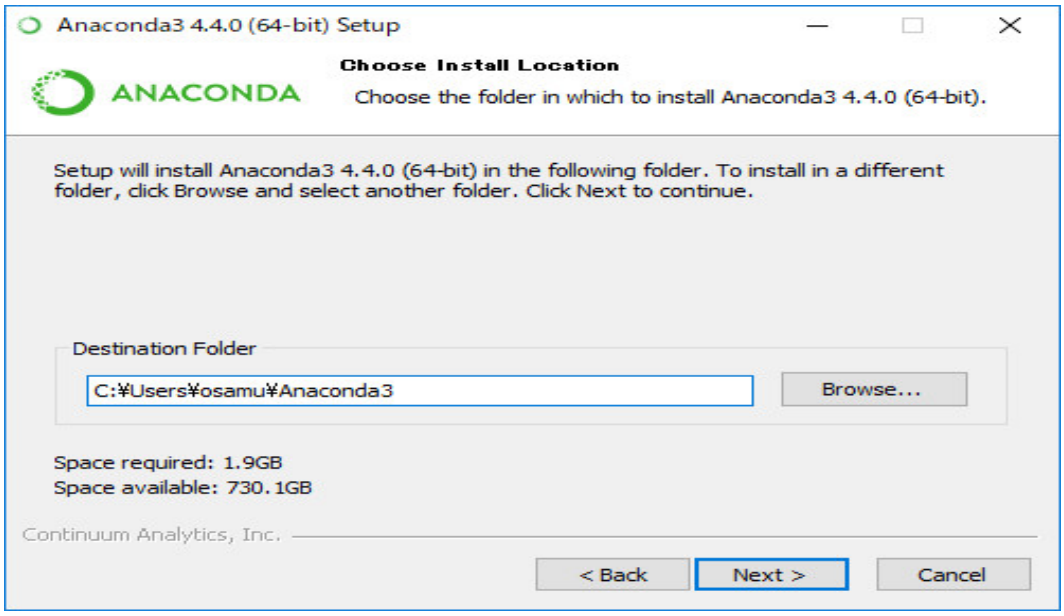
「Next」をクリックします。



「I Agree」をクリックします。



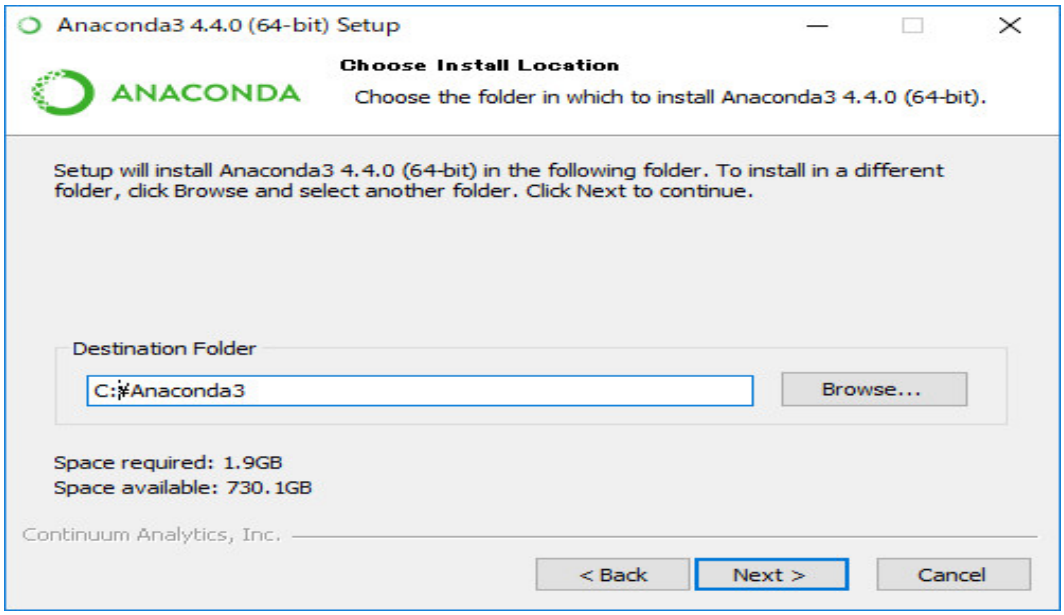
「Next」をクリックします。



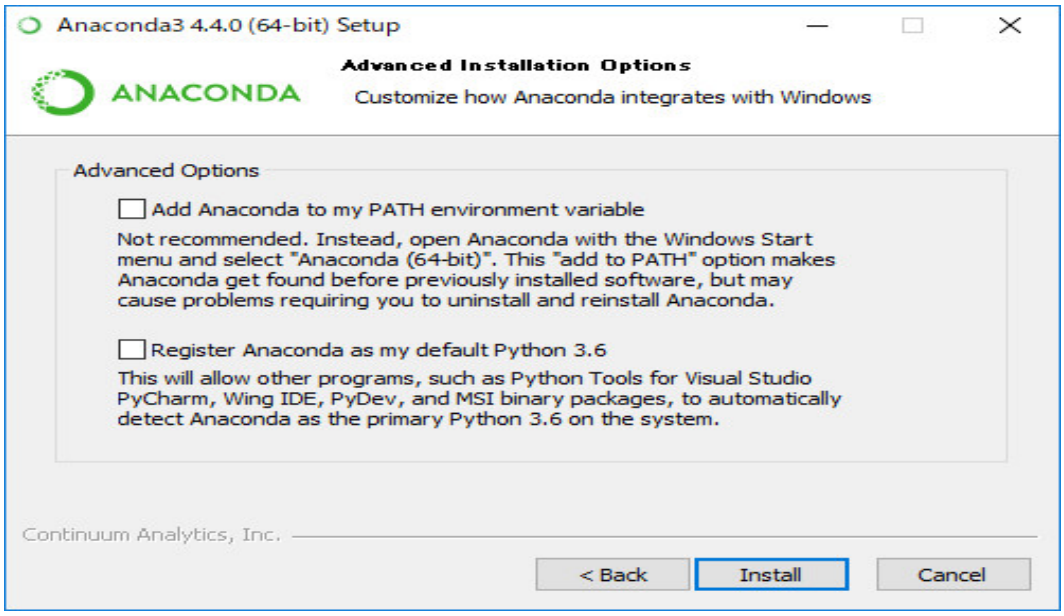
Destination Folder がデフォルトでは、後で色々問題が起こるのでここを

`C:\Anaconda3`

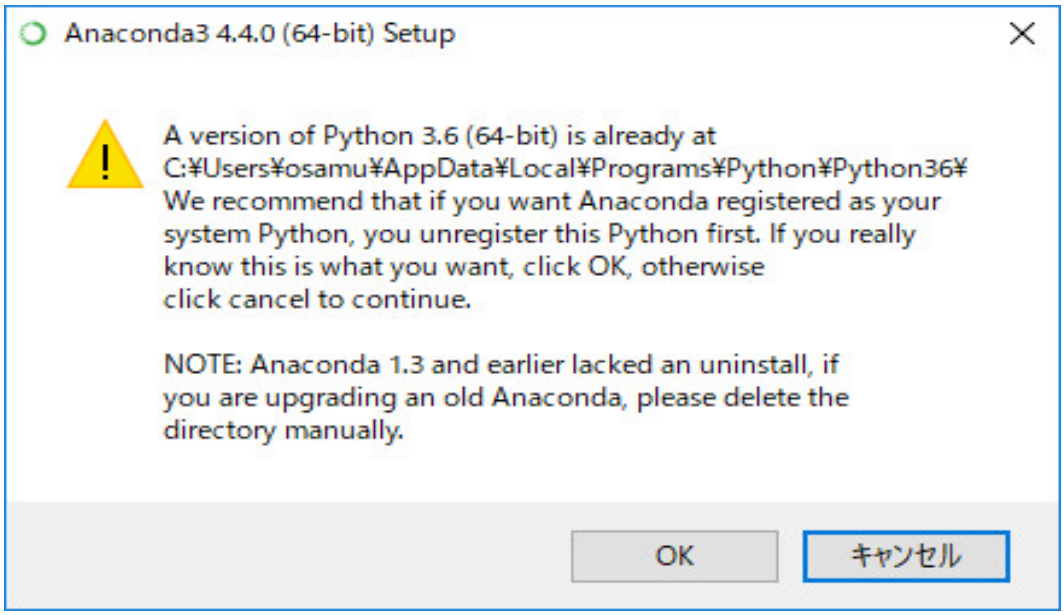
と修正します。



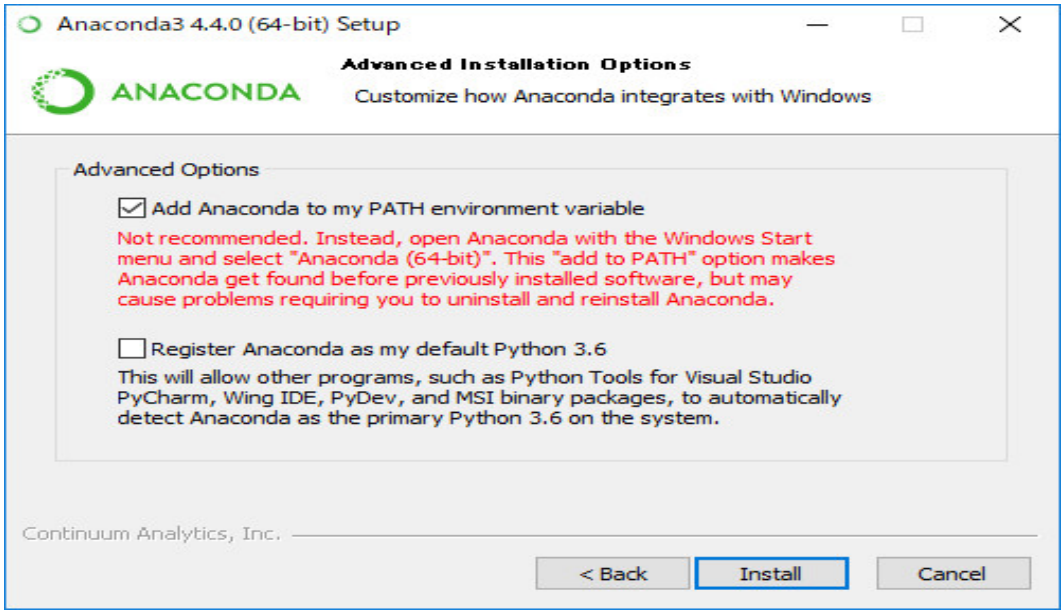
「Next」をクリックします。



通常は両方にチェックを入れます。しかし、すでに Python3.6 をインストールしている場合には、下にチェックを入れると

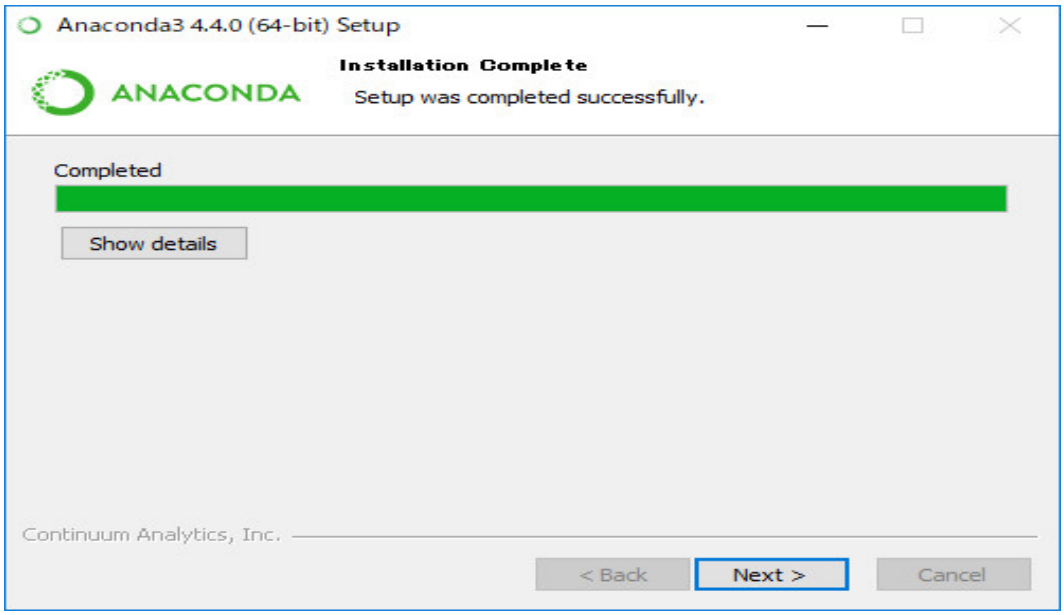


と表示されるので、「キャンセル」をクリックし、すでに Python3.6 をインストールしている場合には

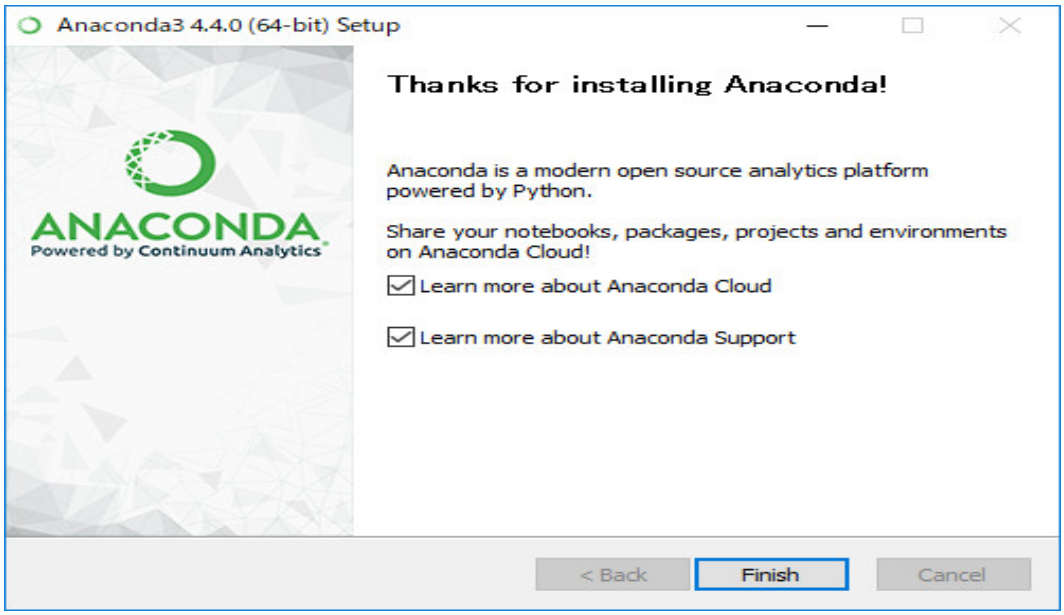


のように、上だけをチェックします。

「Install」をクリックします。

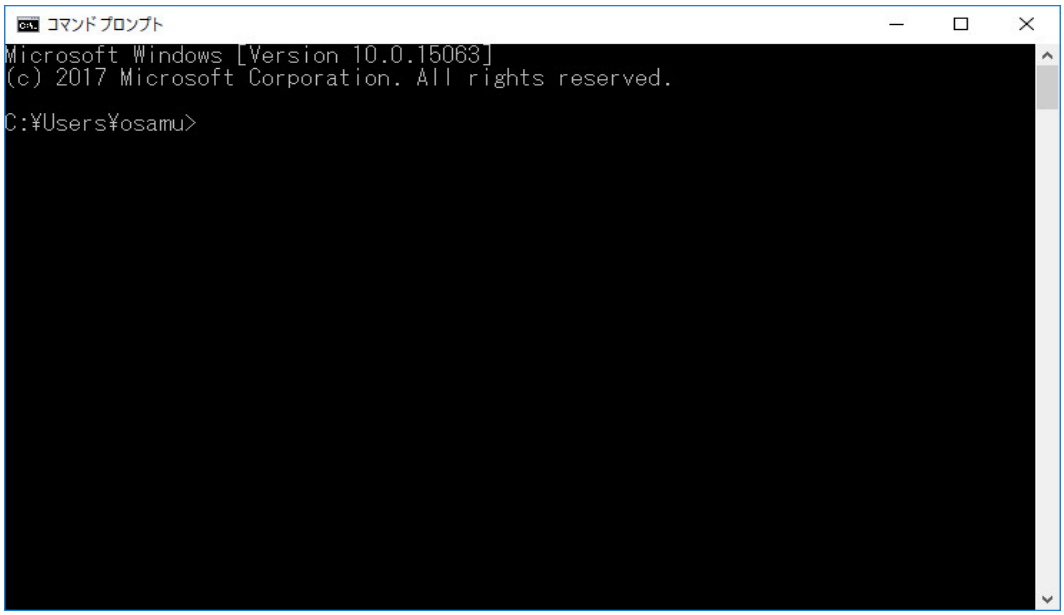


「Next」をクリックします。

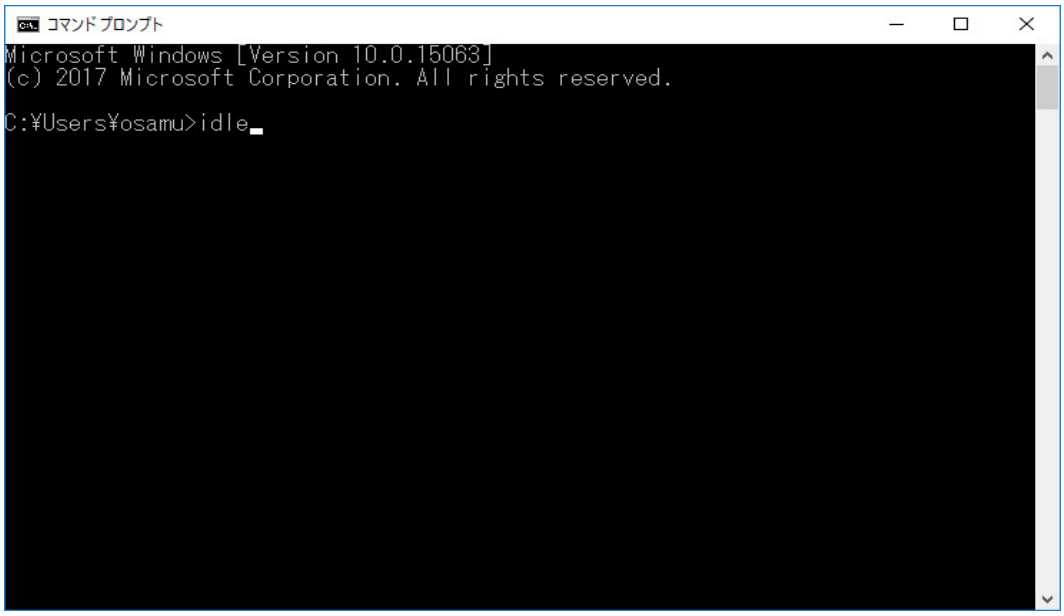


「Finish」をクリックします。

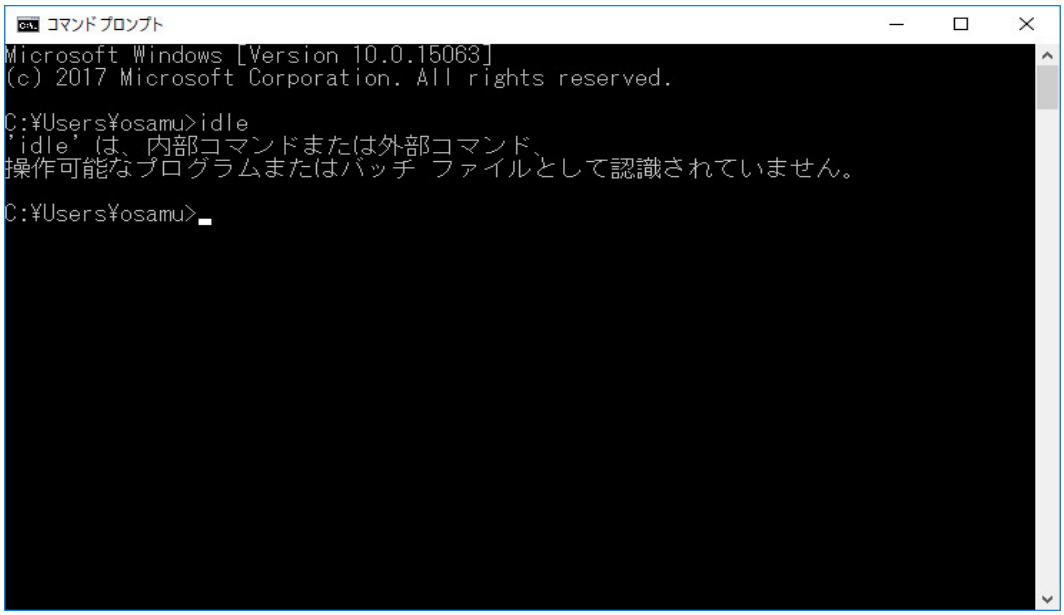
Anaconda をインストールした後に「コマンド　プロンプト」を起動し、


コマンド プロンプト
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.
C:¥Users¥osamu>

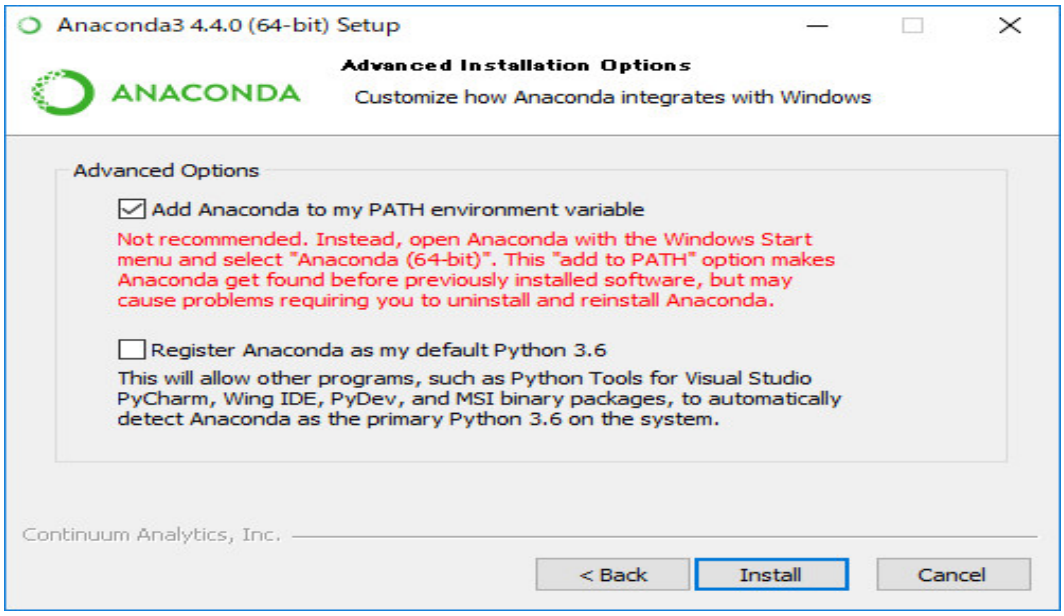
「idle」を


コマンド プロンプト
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.
C:¥Users¥osamu>idle

実行します。

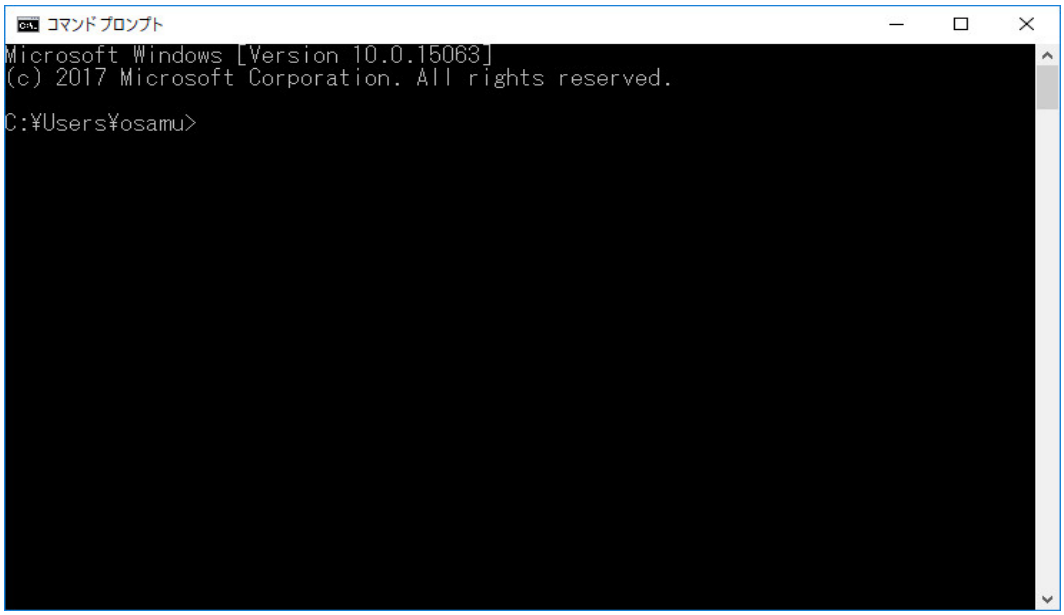


と表示されたら、

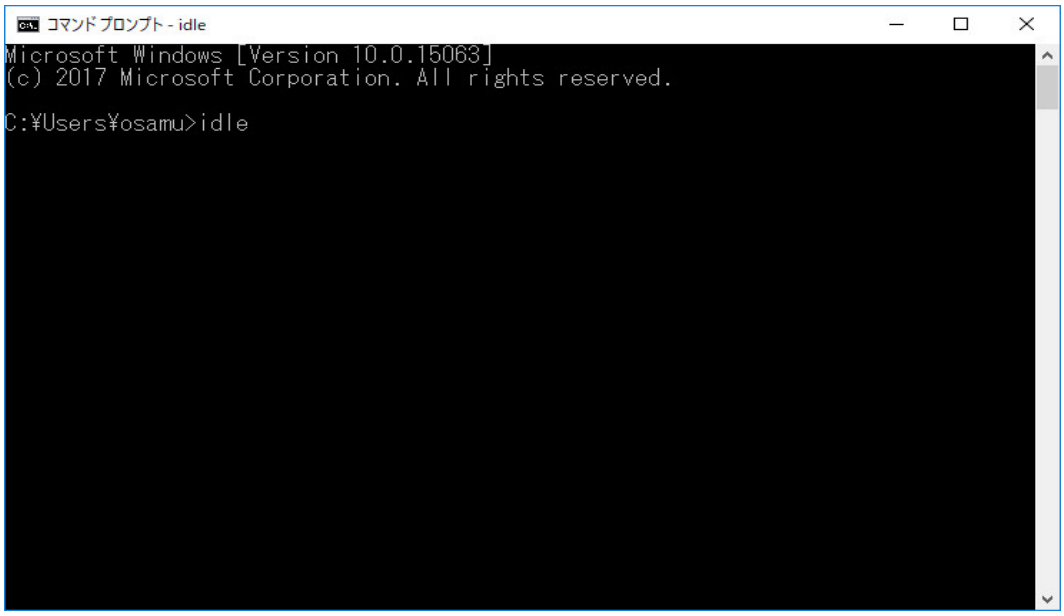


のチェックを忘れていた可能性があります。 Anaconda3 のフォルダーをエクスプローラーで削除して、もう一度インストールをやり直すのが簡単です。

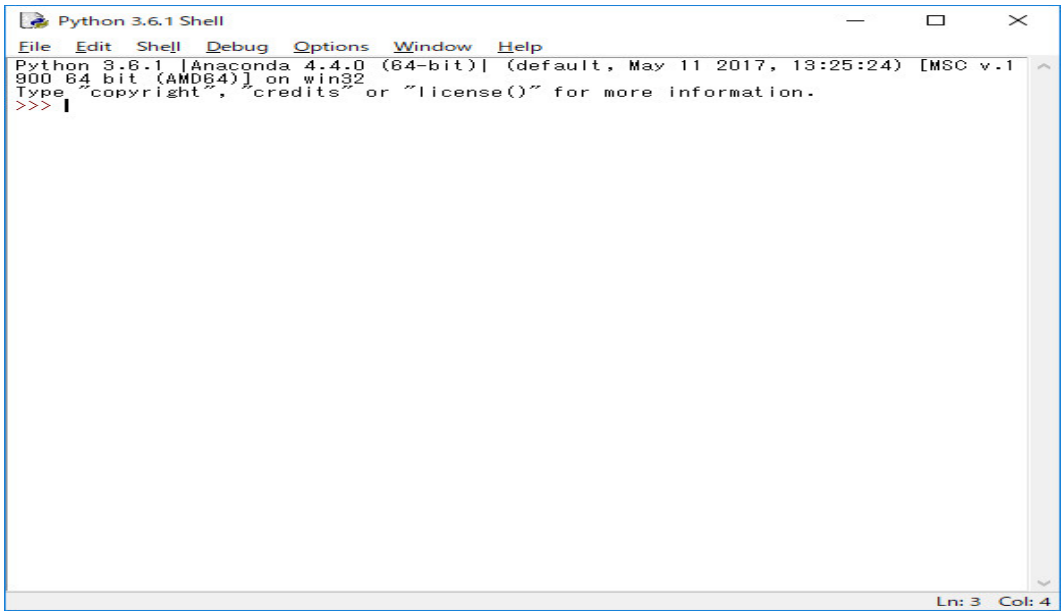
「コマンド　プロンプト」を起動し、


コマンド プロンプト
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.
C:¥Users¥osamu>

「idle」を

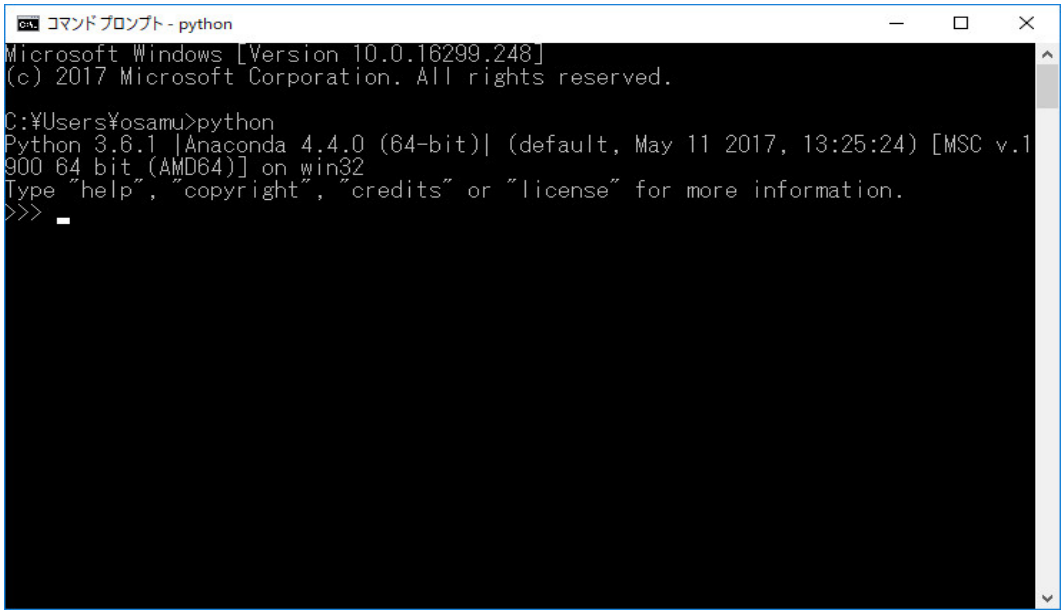

コマンド プロンプト - idle
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.
C:¥Users¥osamu>idle

実行します。



と「Python3.6.1 Shell」が起動したら、Anaconda のインストールは完成です。

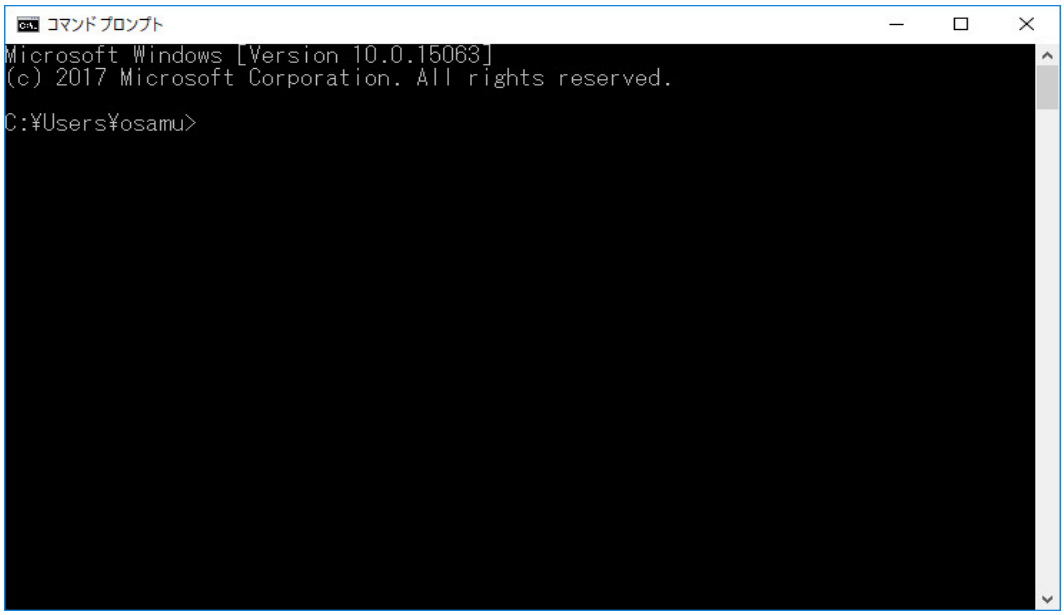
「コマンド　プロンプト」を起動し、「python」を



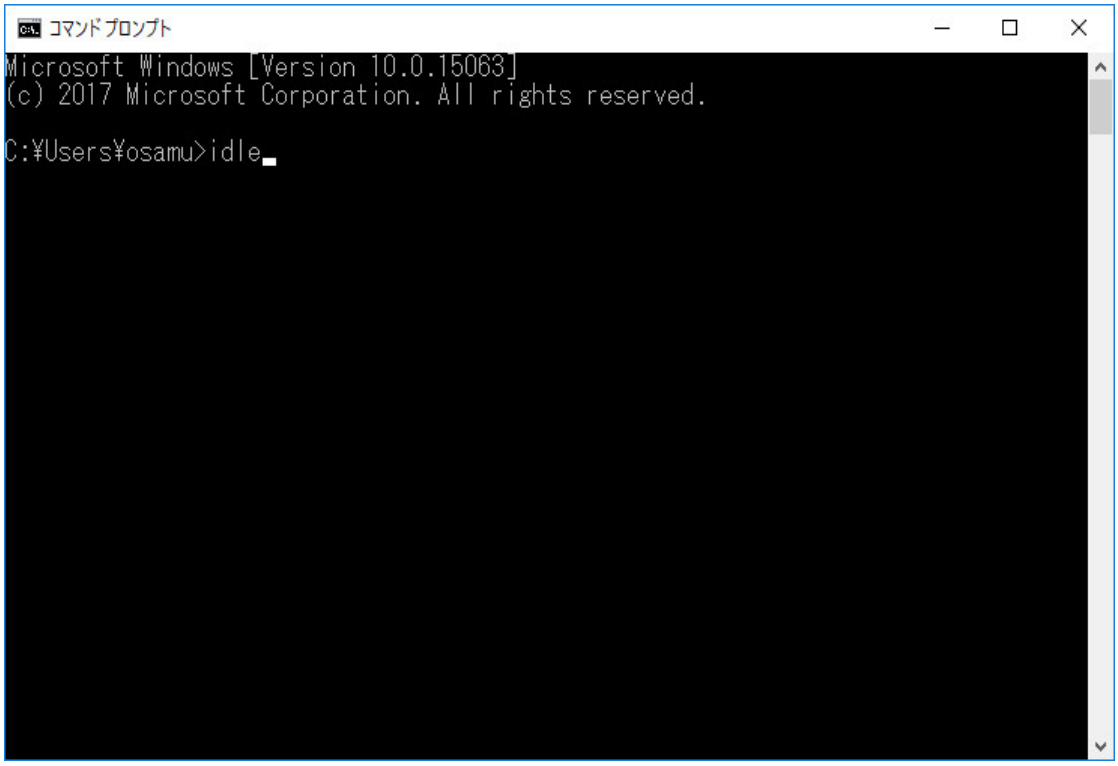
実行すると Python インタープリターが走り始めます。

それでは、初めて **Python** をインストールした学生さんがフラッシュカードのプログラムを作れるように解説していきます。プログラムリストだけが欲しい人のためには、この解説の最後にプログラムリストを提示します。

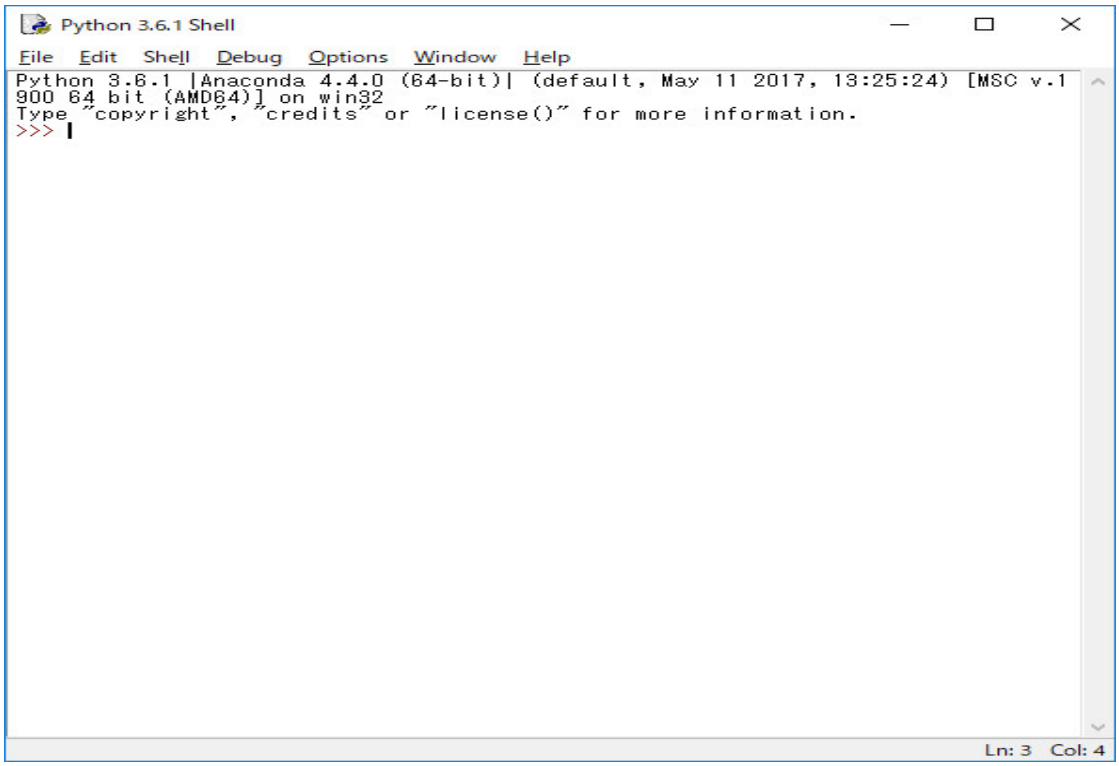
Anaconda をインストールした後に「コマンド　プロンプト」を起動し、


コマンド プロンプト
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.
C:¥Users¥osamu>

「idle」を

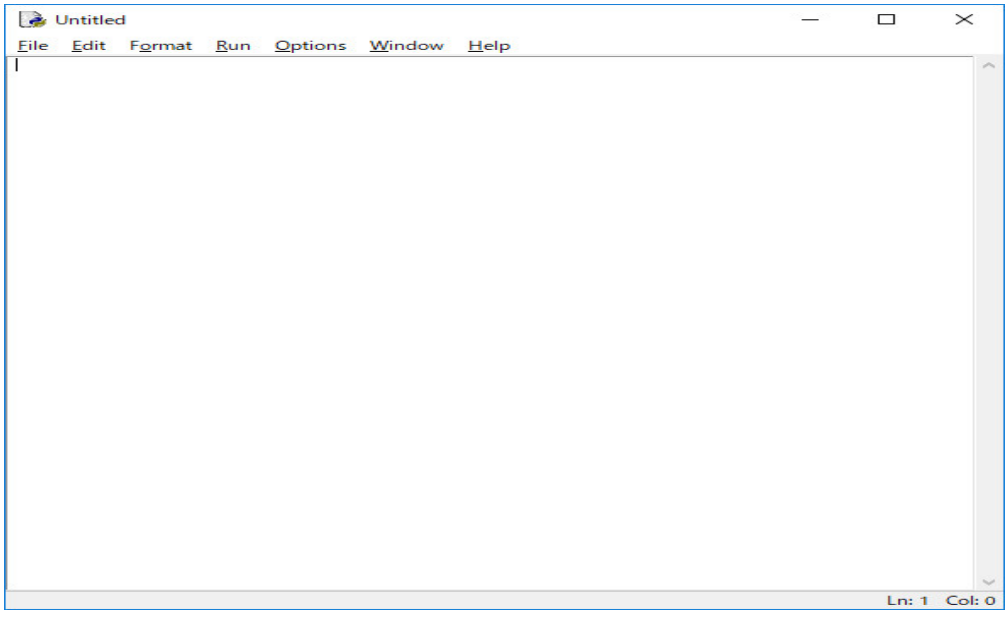

コマンド プロンプト
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.
C:¥Users¥osamu>idle

実行します。



Python の Shell が立ち上がります。

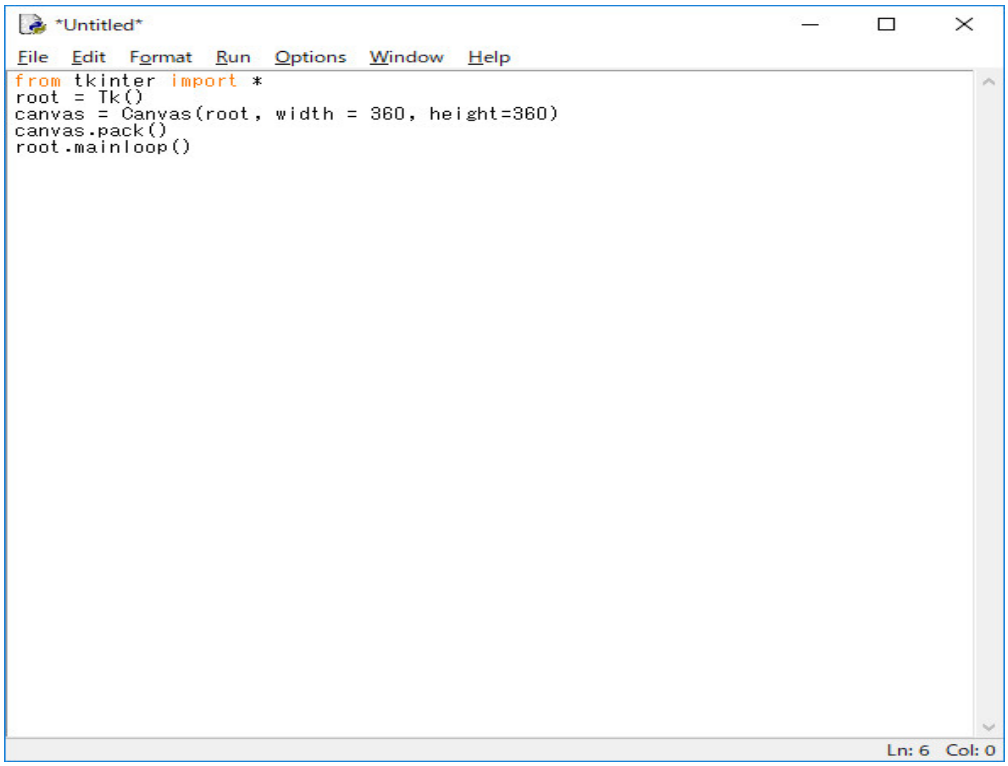
File メニューの　New File を選択する。Editor が開く。



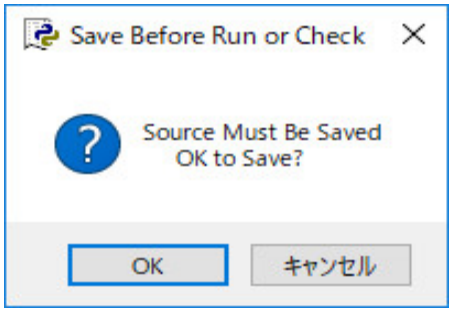
ここに、次のように打ち込む。

```
from tkinter import *
root = Tk()
```

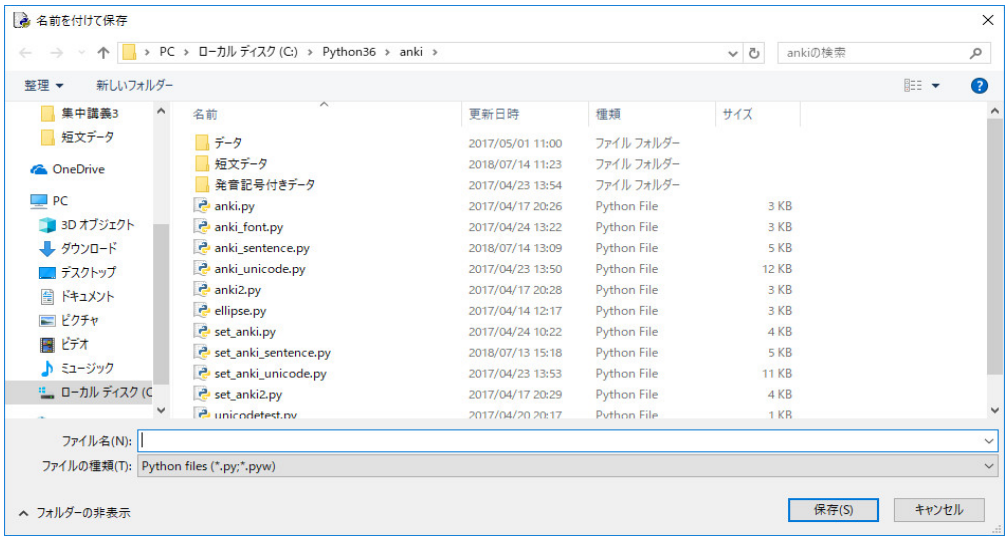
```
canvas = Canvas(root, width = 360, height=360)
canvas.pack()
root.mainloop()
```



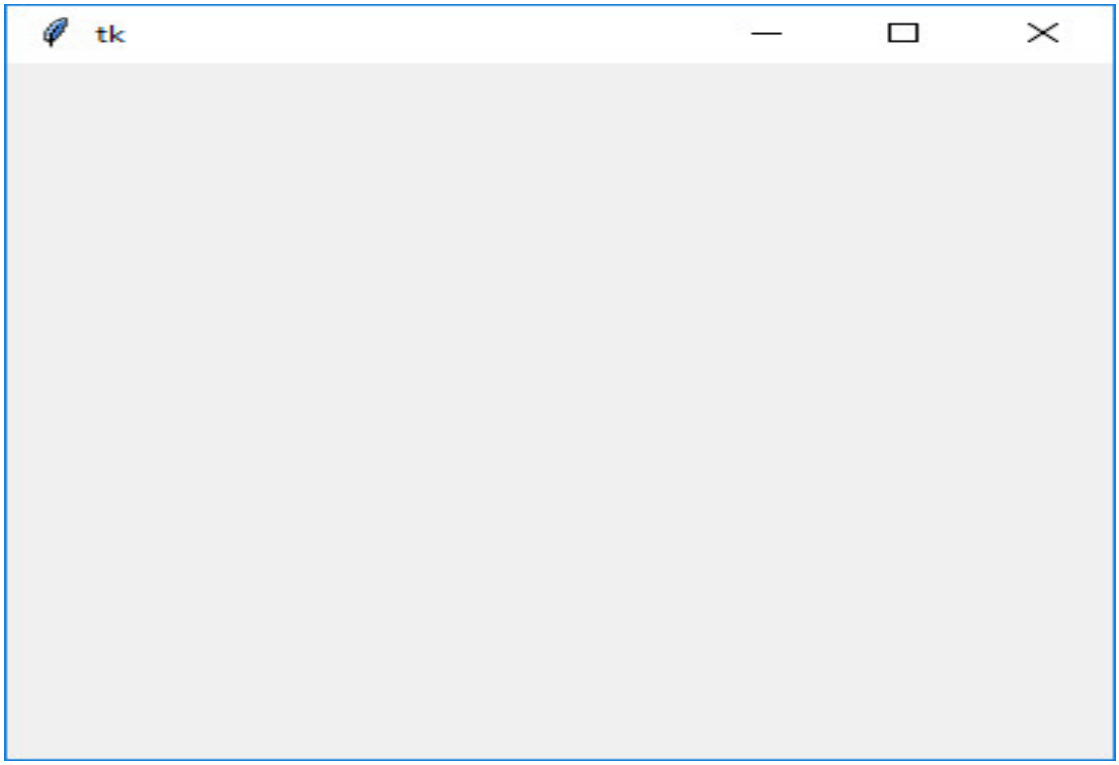
Run メニューの　Run Module を選択する。



OK のボタンをクリックする。



このプルグラムを適当なフォルダに適当な名前（例えば、words.py）を付け保存する。



360 × 360 の空のキャンバスを持ったウインドウが表示される。

```
from tkinter import *
root = Tk()
canvas = Canvas(root, width = 360, height=360)
```

```
canvas.pack()
root.mainloop()
```

は、

```
from tkinter import *
```

で、まず tkinter を import します。tkinter は Python に最初から備わっています。次に

```
root = Tk()
```

で、top level の main window を作り、root という名前を付けています。次に、

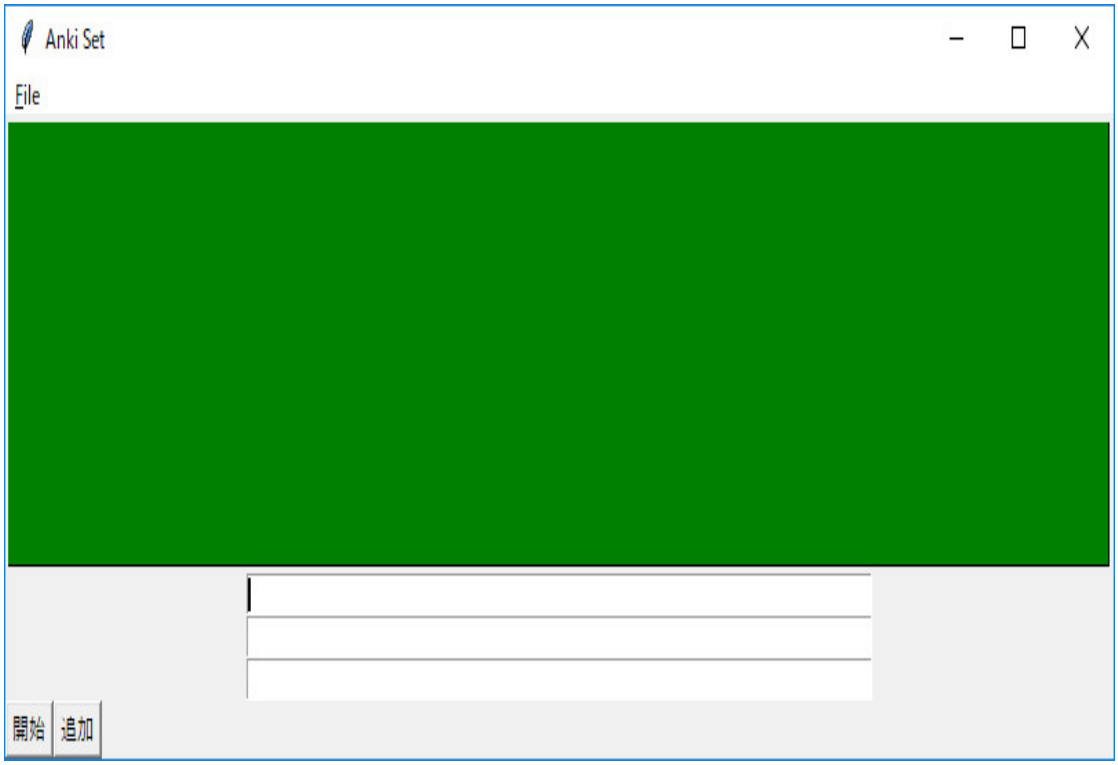
```
canvas = Canvas(root, width = 360, height=360)
canvas.pack()
```

の 2 行で、ウィジェットを設定して window に配置します。今の場合、幅 360、高さ 360 の Canvas を canvas という名前で main window である root に設定配置しています。最後に、

```
root.mainloop()
```

で、event loop を開始して user からの要求 (event:ボタンが押されたとかマウスがクリックされたとか) を処理します。tkinter を使った window プログラミングはいつでもこのようにプログラミングを始めるものだと思って下さい。

まず、フラッシュカードのデータを作るソフトを作ります。最終的には



のようなプログラムを作ります。
　ソフトのタイトルがあり、File というメニューがあり、その中に「保存」と「読込」というサブメニューがあり、画面中央にデータを表示する canvas があり、その下に、英単語と訳と付加情報を入力するための Entry が三個あり、一番下に、「開始」と「追加」と表示された Button があります。
　ソフトのタイトルを表示するには、

```
root.title("Anki Set")
```

とします。Canvas のサイズは、大きければ沢山の情報を表示できますが、自分のパソコンで見やすい大きさにしますが、ここでは

```
canvas = Canvas(root, width = 800, height=200)
```

としておきます。エントリーを表示するためには

```
# エントリー
e = Entry(root, width=75,)
e.pack()
e.focus_set()
e2 = Entry(root, width=75)
e2.pack()
e3 = Entry(root, width=75)
e3.pack()
```

のように定義し、Button を表示するには

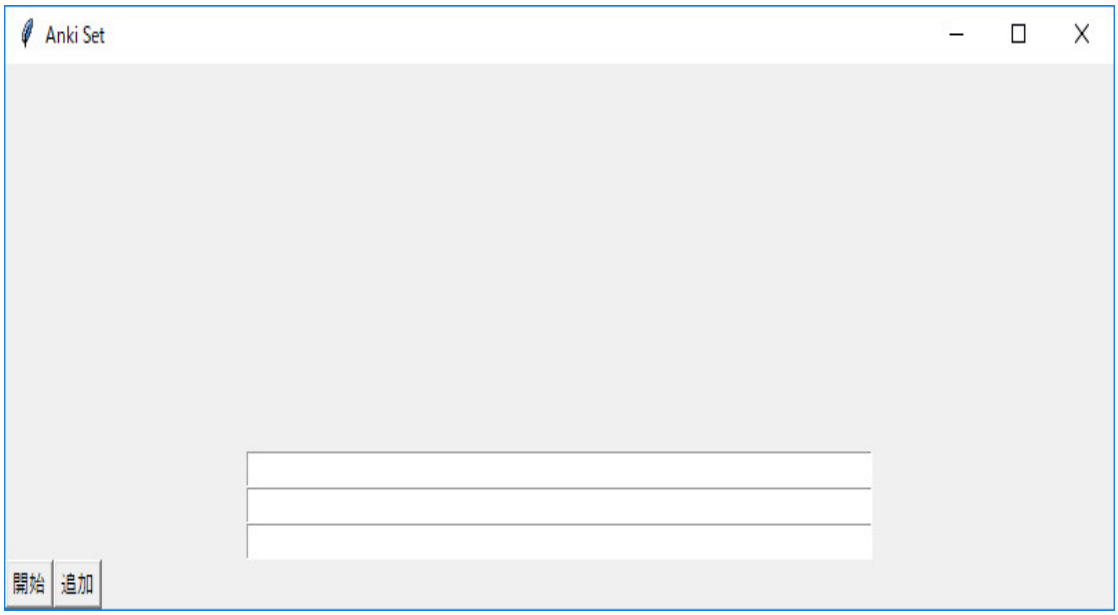
```
button3 = Button(root, text = '開始')
button3.pack(side=LEFT)
button1 = Button(root, text = '追加')
button1.pack(side=LEFT)
```

とします。メニューは後回しにします。
　プログラムは

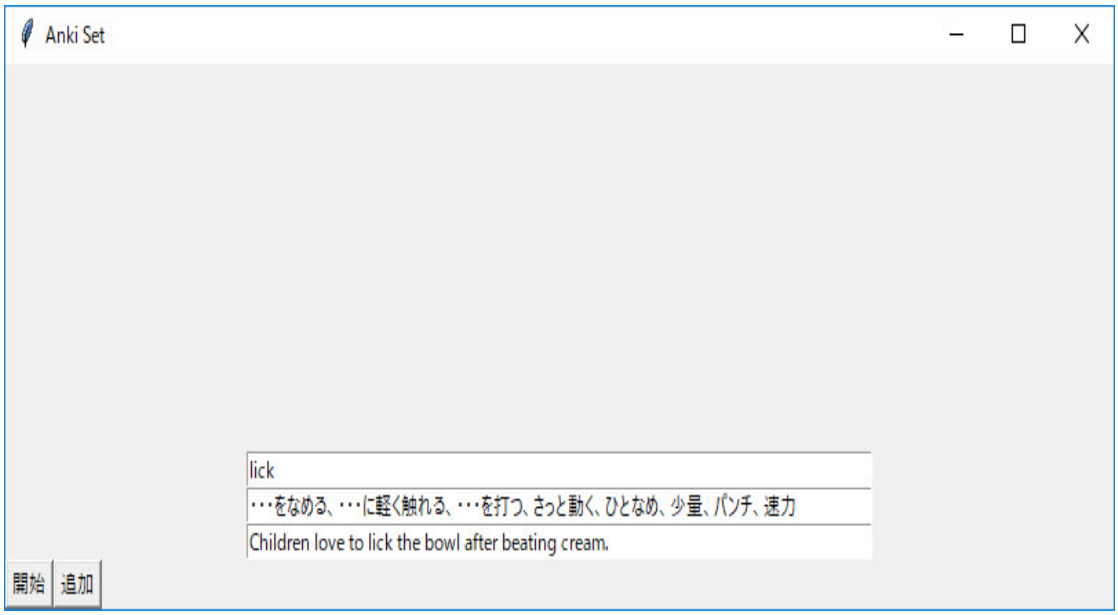
```
from tkinter import *
root = Tk()
root.title("Anki Set")
canvas = Canvas(root, width = 800, height=200)
canvas.pack()
e = Entry(root, width=75)
e.pack()
```

```
e.focus_set()
e2 = Entry(root, width=75)
e2.pack()
e3 = Entry(root, width=75)
e3.pack()
button3 = Button(root, text = '開始')
button3.pack(side=LEFT)
button1 = Button(root, text = '追加')
button1.pack(side=LEFT)
root.mainloop()
```

となります。実行すると



です。



のように、エントリーに文字を入力することは出来ます。この文字列を取り出し、次々リストにセットできるようにします。リストは、例えば、[”exploit”, ” … を搾取する”, ”exploit ... to the full （ … を最大限利用する）”] のように文字列や数値をコンマで区切り、並べたものを [と] で囲ったものです。

まず、エントリーの定義を

```
# エントリー
e = Entry(root, width=75, textvariable = buffer)
e.pack()
e.focus_set()
e2 = Entry(root, width=75, textvariable = buffer2)
e2.pack()
e3 = Entry(root, width=75, textvariable = buffer3)
e3.pack()
```

と修正し、textvariable にエントリーの値にアクセスするための変数をセットし、その変数を

```
# エントリーの値を格納するオブジェクト
buffer = StringVar()
buffer.set("")
buffer2 = StringVar()
buffer2.set("")
buffer3 = StringVar()
buffer3.set("")
```

と定義します。
　エントリーに文字列を入力し、Enter キーを押せば、キャンバスに表示するようにします。

```
# 単語と意味と短文を格納する変数
word = ""
value = ""
sentence = ""
```

と定義し、Enter キーを押したとき実行する関数を

```
# バインディング
e.bind('<Return>', set_word)
e2.bind('<Return>', set_value)
e3.bind('<Return>', set_sentence)
```

とセットし、関数 set_word(event) を

```
def set_word(event):
    global word
    if buffer.get():
        word = buffer.get()
        canvas.create_text(400, 35, text=word, font=('FixedSys', 14))
        e2.focus_set()
```

と定義し、関数 set_value(event) を

```
def set_value(event):
    global value
    if buffer2.get():
        value = buffer2.get()
        canvas.create_text(400, 105, text=value, font=('FixedSys', 14))
        e3.focus_set()
```

と定義し、関数 set_sentence(event) を

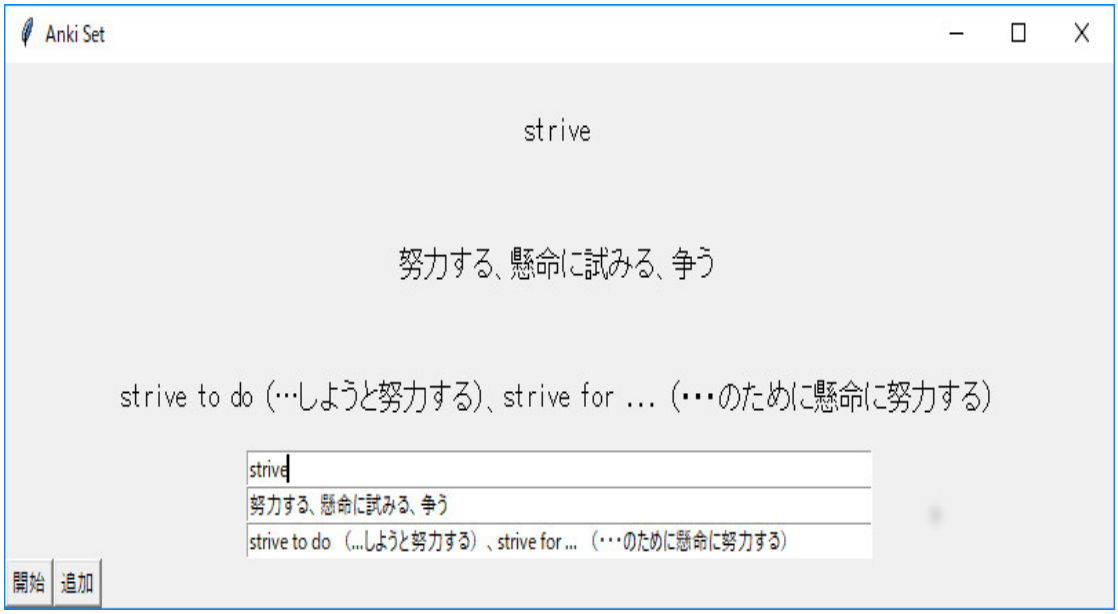
```
def set_sentence(event):
    global sentence
    if buffer3.get():
        sentence = buffer3.get()
        canvas.create_text(400, 175, text=sentence, font=('FixedSys', 14))
        e.focus_set()
```

と定義します。ここまでで、プログラムは

```
from tkinter import *

root = Tk()
root.title("Anki Set")
canvas = Canvas(root, width = 800, height=200)
canvas.pack()

# 単語と意味と短文を格納する変数
word = ""
value = ""
sentence = ""

# 表示
def set_word(event):
    global word
    if buffer.get():
        word = buffer.get()
        canvas.create_text(400, 35, text=word, font=('FixedSys', 14))
        e2.focus_set()

def set_value(event):
    global value
    if buffer2.get():
        value = buffer2.get()
        canvas.create_text(400, 105, text=value, font=('FixedSys', 14))
        e3.focus_set()
def set_sentence(event):
    global sentence
    if buffer3.get():
        sentence = buffer3.get()
        canvas.create_text(400, 175, text=sentence, font=('FixedSys', 14))
        e.focus_set()

# エントリーの値を格納するオブジェクト
buffer = StringVar()
buffer.set("")
buffer2 = StringVar()
buffer2.set("")
buffer3 = StringVar()
```

```
buffer3.set("")
# エントリー
e = Entry(root, width=75, textvariable = buffer)
e.pack()
e.focus_set()
e2 = Entry(root, width=75, textvariable = buffer2)
e2.pack()
e3 = Entry(root, width=75, textvariable = buffer3)
e3.pack()
# バインディング
e.bind('<Return>', set_word)
e2.bind('<Return>', set_value)
e3.bind('<Return>', set_sentence)

button3 = Button(root, text = '開始')
button3.pack(side=LEFT)
button1 = Button(root, text = '追加')
button1.pack(side=LEFT)

root.mainloop()
```

となります。実行すると



です。

次に、ボタン「開始」をクリックすれば、単語、訳、短文を保持する変数を、空のリストに初期化するようにします。

```
button3 = Button(root, text = '開始', command = start)
button3.pack(side=LEFT)
button1 = Button(root, text = '追加', command = add_word)
button1.pack(side=LEFT)
```

のように修正すれば、command にセットした関数が、そのボタンをクリックしたときに実行されます。今の場合、「開始」のボタンをクリックすれば、start() という関数が、「追加」のボタンをクリックすれば、add_word() という関数が実行されます。まず、

```
words_list = []
sentences_list = []
answers_list = []
```

とグローバル変数（関数の中で定義されたローカル変数でない変数）としてリストを保持する変数を定義し、空に初期化しておきます。関数 start() を

```
def start():
    global words_list, sentences_list, answers_list

    words_list = []
    sentences_list = []
    answers_list = []

    canvas.create_rectangle(0, 0, 800, 200, fill='green')
```

と定義します。

```
    global words_list, sentences_list, answers_list
```

というグローバル変数であるという宣言なしに、

```
def start():
    words_list = []
    sentences_list = []
    answers_list = []

    canvas.create_rectangle(0, 0, 800, 200, fill='green')
```

と定義すると変数 words_list, sentences_list, answers_list がローカル変数（関数 start() の中だけで有効な変数）と理解され、折角初期化しても、関数 start() の終了が終わるとグローバル変数の値は初期化されていず、元の値のままです。

```
    canvas.create_rectangle(0, 0, 800, 200, fill='green')
```

は Canvas に Canvas の大きさの矩形を描いて、緑で塗りつぶしています。white の方がいいかもわかりません。

```
canvas.create_rectangle(0, 0, 800, 200, fill='green')
```

を関数の外にも書いておけば、ソフトが立ち上がった時に Canvas が緑で塗りつぶされます。

関数 add_word() を

```
def add_word():
    global word, value, sentence, words_list, sentences_list, answers_list

    words_list.append(word)
    answers_list.append(value)
    sentences_list.append(sentence)
    canvas.create_rectangle(0, 0, 800, 200, fill='white')
    buffer.set("")
    buffer2.set("")
    buffer3.set("")
```

と定義します。

```
    words_list.append(word)
    answers_list.append(value)
    sentences_list.append(sentence)
```

で、word, value, sentence の値を変数 words_list, sentences_list, answers_list の最後にそれぞれ追加しています。

```
    canvas.create_rectangle(0, 0, 800, 200, fill='white')
    buffer.set("")
    buffer2.set("")
    buffer3.set("")
```

で、Canvas を消し、エントリーを空にしています。ここまでのプログラムは

```
from tkinter import *

root = Tk()
root.title("Anki Set")
canvas = Canvas(root, width = 800, height=200)
canvas.pack()

# 単語と意味と短文を格納する変数
word = ""
value = ""
sentence = ""
words_list = []
sentences_list = []
answers_list = []

def start():
    global words_list, sentences_list, answers_list

    words_list = []
    sentences_list = []
    answers_list = []

    canvas.create_rectangle(0, 0, 800, 200, fill='green')

def add_word():
    global word, value, sentence, words_list, sentences_list, answers_list

    words_list.append(word)
    answers_list.append(value)
    sentences_list.append(sentence)
    canvas.create_rectangle(0, 0, 800, 200, fill='white')
    buffer.set("")
    buffer2.set("")
    buffer3.set("")

# 表示
def set_word(event):
    global word
    if buffer.get():
```

```
        word = buffer.get()
        canvas.create_text(400, 35, text=word, font=('FixedSys', 14))
        e2.focus_set()

def set_value(event):
    global value
    if buffer2.get():
        value = buffer2.get()
        canvas.create_text(400, 105, text=value, font=('FixedSys', 14))
        e3.focus_set()
def set_sentence(event):
    global sentence
    if buffer3.get():
        sentence = buffer3.get()
        canvas.create_text(400, 175, text=sentence, font=('FixedSys', 14))
        e.focus_set()

# エントリーの値を格納するオブジェクト
buffer = StringVar()
buffer.set("")
buffer2 = StringVar()
buffer2.set("")
buffer3 = StringVar()
buffer3.set("")
# エントリー
e = Entry(root, width=75, textvariable = buffer)
e.pack()
e.focus_set()
e2 = Entry(root, width=75, textvariable = buffer2)
e2.pack()
e3 = Entry(root, width=75, textvariable = buffer3)
e3.pack()
# バインディング
e.bind('<Return>', set_word)
e2.bind('<Return>', set_value)
e3.bind('<Return>', set_sentence)

canvas.create_rectangle(0, 0, 800, 200, fill='green')
button3 = Button(root, text = '開始', command = start)
```

```
button3.pack(side=LEFT)
button1 = Button(root, text = '追加', command = add_word)
button1.pack(side=LEFT)

root.mainloop()
```

となります。
　次の格納した変数 words_list, sentences_list, answers_list のデータをファイルに保存できるようにします。まず、メニューを表示するようにします。

```
menubar = Menu(root)
root.configure(menu = menubar)
menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "保存", under = 0, command = joseki2file)
menufile.add_command(label = "読込", under = 0, command = file2joseki)
```

のように定義します。こういう風に定義するものだと思ってください。「保存」をクリックすると関数 joseki2file() が実行され、「読込」をクリックすると関数 file2joseki() が実行されます。変な名前ですが気にしないでください。「囲碁の定石」のためのプログラムを流用しています。
　関数 joseki2file() は

```
import tkinter.filedialog
import sys, os.path
```

と import し、

```
path_name = ""
file_name = ""

M_STOP_FLAG = False
```

と変数を定義し、

```
def joseki2file():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
```

```
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
        ss = "[\n"
        f.write(ss)
        for ind in range(len(words_list)):
            w = words_list[ind]
            v = answers_list[ind]
            u = sentences_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
    print('words_dict=', words_list, answers_list, sentences_list)
```

と定義します。

```
    print('words_dict=', words_list, answers_list, sentences_list)
```

はデバッグの為ですので、必要無くなれば、コメント # でコメントアウトしてください。と言うより、書き込むデータが大きくなるとこの部分があるとソフトの実行に時間が掛かるようになり、トラブルの原因になるので、

```
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

のように、最後のデータだけを表示するようにするのが良いです。

このプログラムが気になってしょうがないなら、インターネットで調べてください。こうすればうまくいきます。「読込」をクリックすると関数 file2joseki() が実行されますが、これは作りかけのデータにデータを追加するために、ファイルを読み込んでデータを追加できるようにするためのものです。関数 file2joseki() は

```
def file2joseki():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
```

```
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        words_list = []
        sentences_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
    print('words_dict=', words_list, answers_list, sentences_list)
```

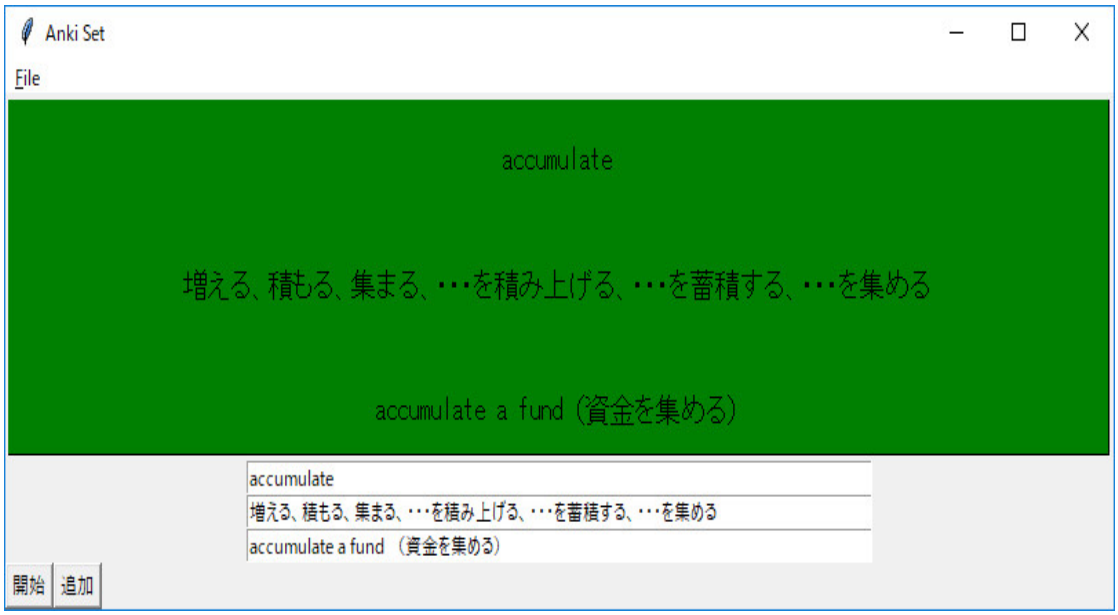
と定義します。

```
    print('words_dict=', words_list, answers_list, sentences_list)
```

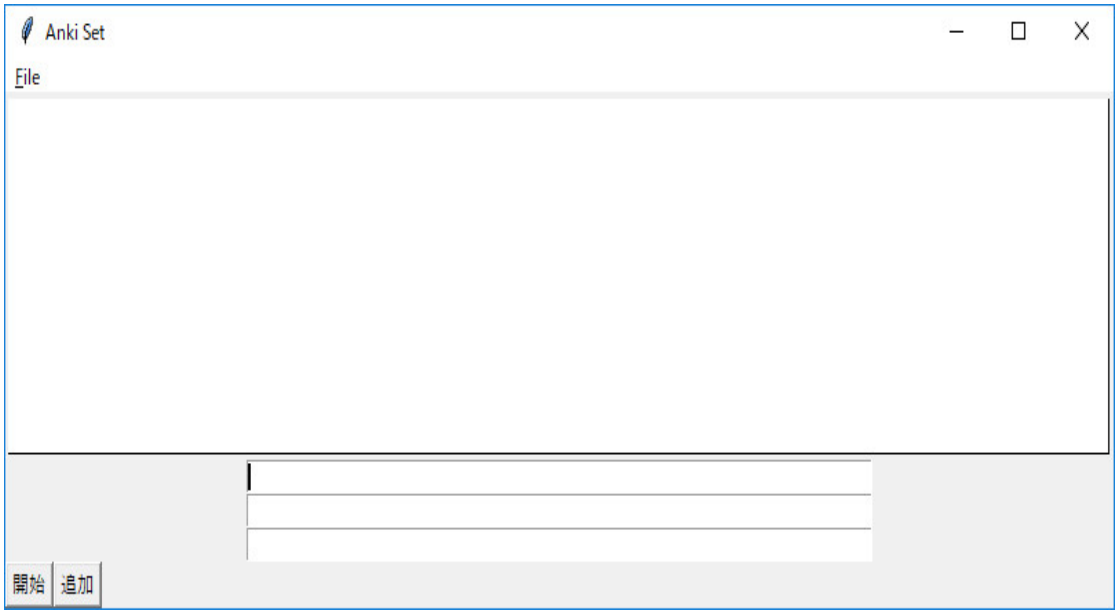
はデバッグの為ですので、必要無くなれば、コメント # でコメントアウトしてください。ここも、実際にこのソフトを使う時は

```
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

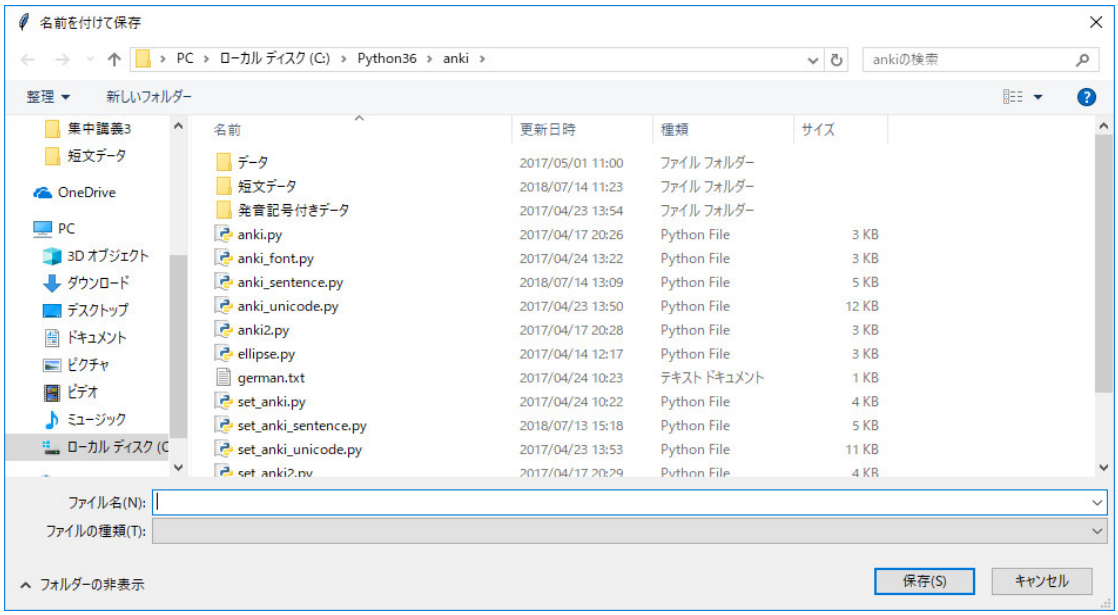
のように、データの最後だけ表示するように修正します。
　実行し、



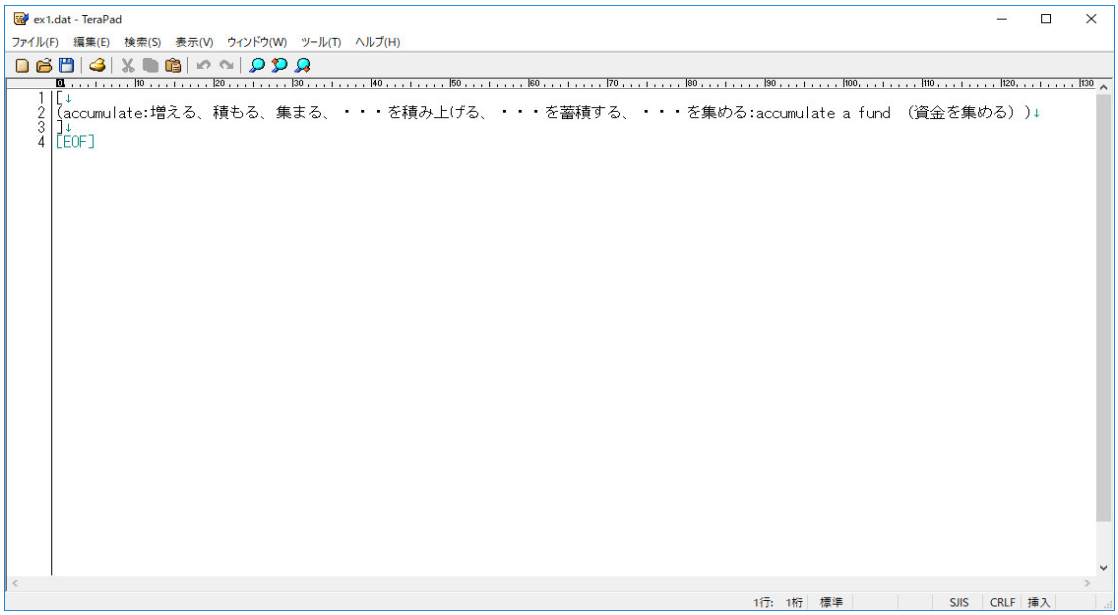
のようにデータを入力し、「追加」のボタンをクリックし、



メニューの「保存」をクリックします。



ファイル名を入力し、データを保存します。データは



のように保存されています。一組のデータが

(accumulate:増える、積もる、集まる、… を積み上げる、… を蓄積する、… を集める:accumulate a fund （資金を集める）)

のように：で区切られて、() で囲まれて、一行で表現されています。これらのデータを [] で囲っています。[と] は一行に一個です。従って、このファイルは適当なエディタで作ることも修正することもできます。

注意：

```
f = open(filename, "w")
```

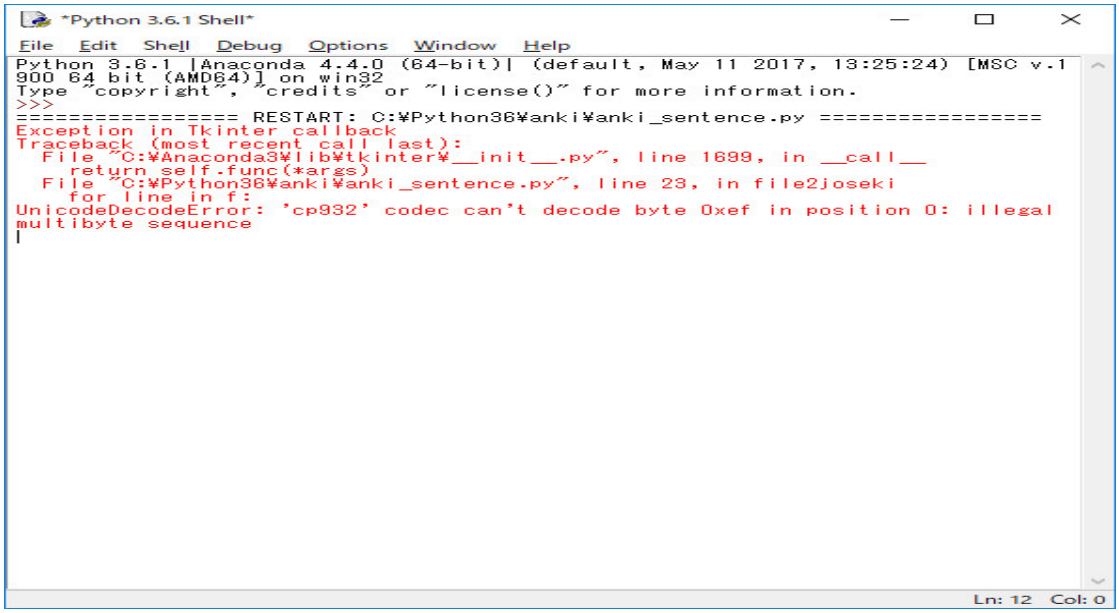
と

```
f = open(filename, "r")
```

でも、このプログラムでは困りませんが、

```
f = open(filename, "w")
```

とすると、エディタでデータを作って、**utf-8** の文字コードで保存したファイルを読み込むと



のようなエラー表示となります。**Python** では、**Windows 10** でファイルに読み書きする時、特殊な文字コード **cp932** を使っています。それで、

```
def joseki2file():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
```

```
        ss = "[\n"
        f.write(ss)
        for ind in range(len(words_list)):
            w = words_list[ind]
            v = answers_list[ind]
            u = sentences_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

のように

```
        f = open(filename, "w", encoding='utf-8')
```

を使い、、

```
def file2joseki():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        words_list = []
        sentences_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
```

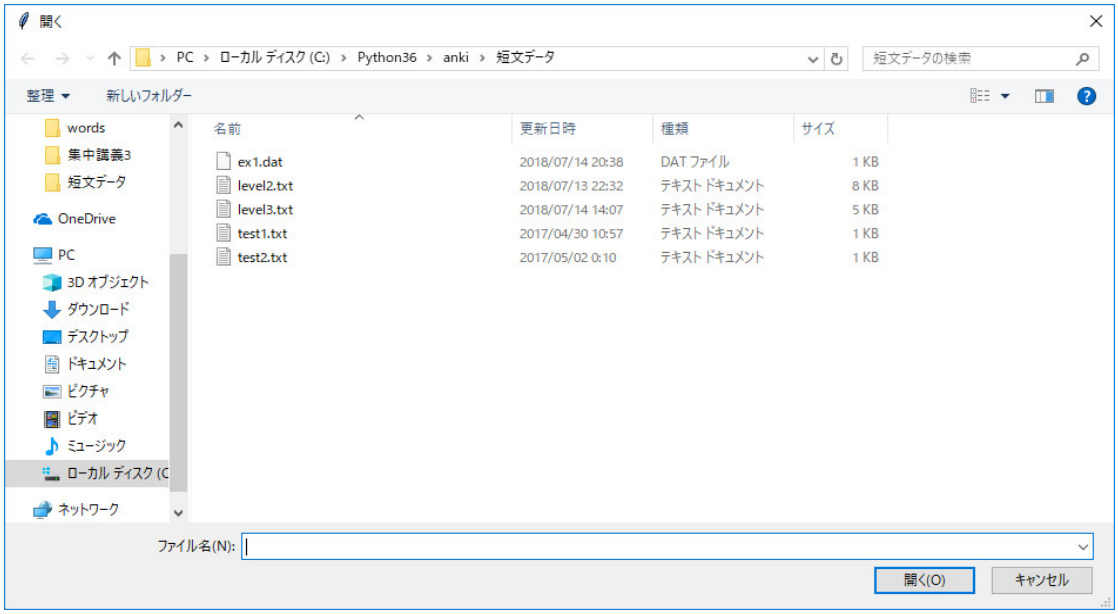
```
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

のように

```
        f = open(filename, "r", encoding='utf-8')
```

を使う方が良いです。そして、エディタでは、デフォルトの文字コード **SHIFT-JIS** ではなく、文字コードは **utf-8** で保存します。更に、データには半角の [と] は使わないようにします。現在は色々な文字コードが使われていて、複雑怪奇です。

このファイルにデータを追加したい時には、メニューの「読込」をクリックして、



データを読み込めば、変数 `words_list`, `sentences_list`, `answers_list` にデータがセットされます。その時、「開始」のボタンをクリックしないように注意してください。クリックすると、変数 `words_list`, `sentences_list`, `answers_list` が空リストに初期化されてしまいます。プログラムの全体は

```
from tkinter import *
import tkinter.filedialog
import sys, os.path

root = Tk()
root.title("Anki Set")
canvas = Canvas(root, width = 800, height=200)
canvas.pack()

path_name = ""
file_name = ""

# 単語と意味と短文を格納する変数
word = ""
value = ""
sentence = ""
words_list = []
sentences_list = []
answers_list = []

M_STOP_FLAG = False

def joseki2file():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
        ss = "[\n"
        f.write(ss)
        for ind in range(len(words_list)):
            w = words_list[ind]
            v = answers_list[ind]
            u = sentences_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            f.write(ss)
```

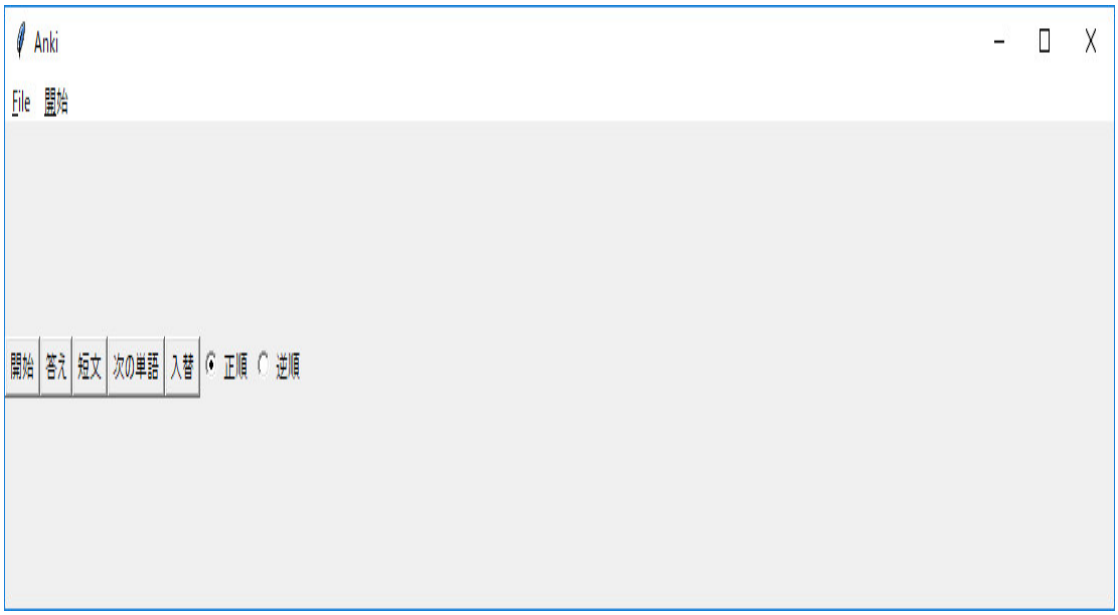
```
            ss = "]\n"
            f.write(ss)
            f.close()
        M_STOP_FLAG = False
        print('words_dict=', words_list, answers_list, sentences_list)

    def file2joseki():
        global M_STOP_FLAG, words_list, sentences_list, answers_list

        M_STOP_FLAG = True
        global path_name, joseki
        filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
        if filename:
            path_name = os.path.dirname(filename)
            f = open(filename, "r", encoding='utf-8')
            words_list = []
            sentences_list = []
            answers_list = []
            for line in f:
                line = line[:-1]
                if line.count('[')  or line.count(']'): continue
                j_list = line.split(":")
                w = j_list[0][1:]
                v = j_list[1]
                u = j_list[2][:-1]
                words_list.append(w)
                answers_list.append(v)
                sentences_list.append(u)
            f.close()
        M_STOP_FLAG = False
        print('words_dict=', words_list, answers_list, sentences_list)

    def start():
        global words_list, sentences_list, answers_list

        words_list = []
        sentences_list = []
        answers_list = []
```

```
    canvas.create_rectangle(0, 0, 800, 200, fill='green')

def add_word():
    global word, value, sentence, words_list, sentences_list, answers_list

    words_list.append(word)
    answers_list.append(value)
    sentences_list.append(sentence)
    canvas.create_rectangle(0, 0, 800, 200, fill='white')
    buffer.set("")
    buffer2.set("")
    buffer3.set("")

# 表示
def set_word(event):
    global word
    if buffer.get():
        word = buffer.get()
        canvas.create_text(400, 35, text=word, font=('FixedSys', 14))
        e2.focus_set()

def set_value(event):
    global value
    if buffer2.get():
        value = buffer2.get()
        canvas.create_text(400, 105, text=value, font=('FixedSys', 14))
        e3.focus_set()
def set_sentence(event):
    global sentence
    if buffer3.get():
        sentence = buffer3.get()
        canvas.create_text(400, 175, text=sentence, font=('FixedSys', 14))
        e.focus_set()

# エントリーの値を格納するオブジェクト
buffer = StringVar()
buffer.set("")
buffer2 = StringVar()
buffer2.set("")
```

```
buffer3 = StringVar()
buffer3.set("")
# エントリー
e = Entry(root, width=75, textvariable = buffer)
e.pack()
e.focus_set()
e2 = Entry(root, width=75, textvariable = buffer2)
e2.pack()
e3 = Entry(root, width=75, textvariable = buffer3)
e3.pack()
# バインディング
e.bind('<Return>', set_word)
e2.bind('<Return>', set_value)
e3.bind('<Return>', set_sentence)

menubar = Menu(root)
root.configure(menu = menubar)
menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "保存", under = 0, command = joseki2file)
menufile.add_command(label = "読込", under = 0, command = file2joseki)

canvas.create_rectangle(0, 0, 800, 200, fill='green')
button3 = Button(root, text = '開始', command = start)
button3.pack(side=LEFT)
button1 = Button(root, text = '追加', command = add_word)
button1.pack(side=LEFT)

root.mainloop()
```

となります。

つぎに、上のプログラムで作ったファイルを読み込んで、単語や訳や短文を表示するプログラムを作ります。作るプログラムは

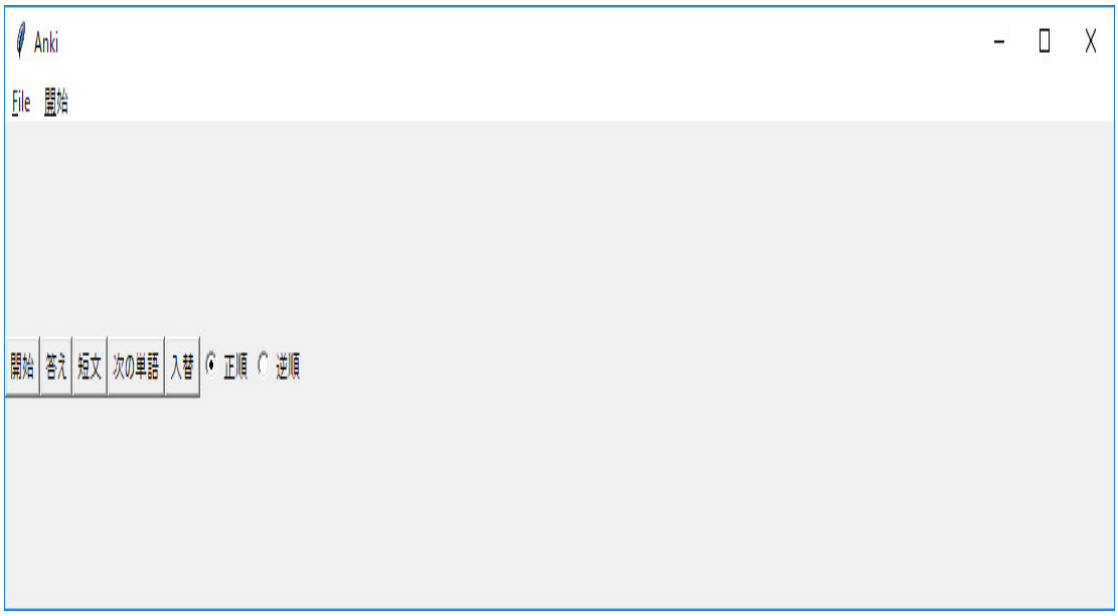


のようなものです。ラジオボタン以外はなじみのあるものです。まず、機能は後回しで、このようなウィジェットを持つウィンドを表示するプログラムを作ります。

```
from tkinter import *
root = Tk()

menubar = Menu(root)
root.configure(menu=menubar)
root.title("Anki")

menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "読込", under = 0)
menubar.add_command(label = '開始', under = 0)

button1 = Button(root, text = '開始')
button1.pack(side=LEFT)

button2 = Button(root, text = '答え')
button2.pack(side=LEFT)

button5 = Button(root, text = '短文')
button5.pack(side=LEFT)

button3 = Button(root, text = '次の単語')
```

```
button3.pack(side=LEFT)

button4 = Button(root, text = '入替')
button4.pack(side=LEFT)

val = IntVar()
val.set(0)
r0 = Radiobutton(root, text = '正順', variable = val, value = 0)
r0.pack(side=LEFT)
r1 = Radiobutton(root, text = '逆順', variable = val, value = 1)
r1.pack(side=LEFT)

canvas = Canvas(root, width=800, height=200)
canvas.pack()
root.mainloop()
```

を実行すると



です。

```
val = IntVar()
val.set(0)
r0 = Radiobutton(root, text = '正順', variable = val, value = 0)
r0.pack(side=LEFT)
r1 = Radiobutton(root, text = '逆順', variable = val, value = 1)
r1.pack(side=LEFT)
```

がラジオボタンをセットしている部分です。このようにすれば上手くいきます。気になるなら、インターネットで調べてください。まず、ファイルからデータを読み込む部分を作ります。これは上で作ったものと同じです。

```
menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "読込", under = 0, command = file2joseki)
menubar.add_command(label = '開始', under = 0, command = start)
```

と修正し、変数を

```
words_list = []
answers_list = []
sentences_list = []

path_name = ""
file_name = ""

index = 0
```

と定義し、関数 file2joseki() を

```
def file2joseki():
    global M_STOP_FLAG, words_list, answers_list, sentences_list
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
```

```
    M_STOP_FLAG = False
    print('words_dict=', words_list, answers_list, sentences_list)
```

と定義します。

データを作るプログラムで

```
def joseki2file():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w")
        ss = "[\n"
        f.write(ss)
        for ind in range(len(words_list)):
            w = words_list[ind]
            v = answers_list[ind]
            u = sentences_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

と

```
def file2joseki():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
```

```
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r")
        words_list = []
        sentences_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

と定義していれば、ここでも

```
def file2joseki():
    global M_STOP_FLAG, words_list, sentences_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r")
        words_list = []
        sentences_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
```

```
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
```

とします。

```
    print('words_dict=', words_list, answers_list, sentences_list)
```

はデバッグの為に入れています。関数 start() は

```
def start():
    global index, words_list, answers_list, sentences_list

    canvas.create_rectangle(0, 0, 800, 200, fill='white')

    index = 0
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
            font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
            font=('FixedSys', 14))
```

と定義します。ここまでのプログラムは

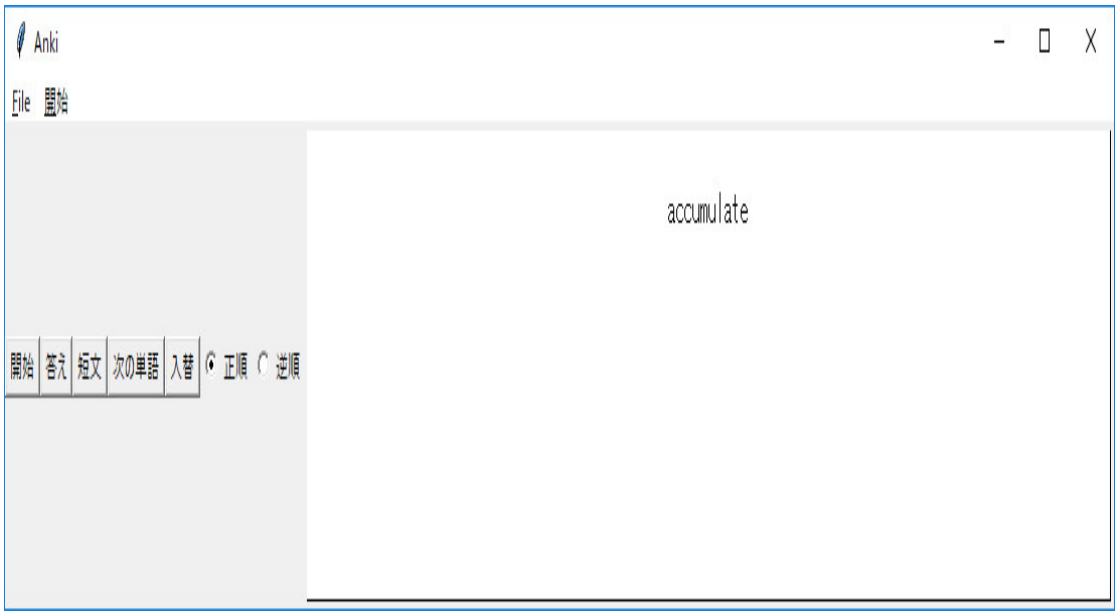
```
from tkinter import *
import tkinter.filedialog
import sys, os.path
```

```
root = Tk()

words_list = []
answers_list = []
sentences_list = []

path_name = ""
file_name = ""

def file2joseki():
    global M_STOP_FLAG, words_list, answers_list, sentences_list
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
    print('words_dict=', words_list, answers_list, sentences_list)

index = 0

def start():
    global index, words_list, answers_list, sentences_list

    canvas.create_rectangle(0, 0, 800, 200, fill='white')

    index = 0
```

```
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
            font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
            font=('FixedSys', 14))

menubar = Menu(root)
root.configure(menu=menubar)
root.title("Anki")

menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "読込", under = 0, command = file2joseki)
menubar.add_command(label = '開始', under = 0, command = start)

button1 = Button(root, text = '開始')
button1.pack(side=LEFT)

button2 = Button(root, text = '答え')
button2.pack(side=LEFT)

button5 = Button(root, text = '短文')
button5.pack(side=LEFT)

button3 = Button(root, text = '次の単語')
button3.pack(side=LEFT)

button4 = Button(root, text = '入替')
button4.pack(side=LEFT)

val = IntVar()
val.set(0)
r0 = Radiobutton(root, text = '正順', variable = val, value = 0)
r0.pack(side=LEFT)
r1 = Radiobutton(root, text = '逆順', variable = val, value = 1)
r1.pack(side=LEFT)
```

```
canvas = Canvas(root, width=800, height=200)
canvas.pack()
root.mainloop()
```

です。実行し、上で作ったファイルを読み込み、「開始」のメニューをクリックすると



です。

```
def start():
    global index, words_list, answers_list, sentences_list

    canvas.create_rectangle(0, 0, 800, 200, fill='white')

    index = 0
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
            font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
            font=('FixedSys', 14))
```

で、ラジオボタンの選択に応じて、words_list か answers_list の最初のデータを表示しています。

次にボタンをクリックしたときの処理を作っていきます。まず、ボタンの定義を

```
button1 = Button(root, text = '開始', command = start)
button1.pack(side=LEFT)

button2 = Button(root, text = '答え', command = show_answer)
button2.pack(side=LEFT)

button5 = Button(root, text = '短文', command = show_sentence)
button5.pack(side=LEFT)

button3 = Button(root, text = '次の単語', command = show_next)
button3.pack(side=LEFT)

button4 = Button(root, text = '入替', command = shuffle)
button4.pack(side=LEFT)
```

と修正します。「開始」のボタンの関数は、メニューの「開始」の関数と同じものを使います。それ以外のそれぞれの関数を

```
def show_answer():
    global index, words_list, answers_list, sentences_list

    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 105, text=answers_list[index],
            font=('FixedSys', 14))
        else:
            canvas.create_text(400, 35, text=words_list[index],
            font=('FixedSys', 14))
```

と

```
def show_sentence():
    global index, words_list, answers_list, sentences_list

    if index < len(words_list):
       canvas.create_text(400, 175, text=sentences_list[index],
       font=('FixedSys', 14))
```

と

```
def show_next():
    global index, words_list, answers_list, sentences_list
```

```
    canvas.create_rectangle(0, 0, 800, 200, fill='white')

    index += 1
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
            font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
            font=('FixedSys', 14))

    else:
        index = 0
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
            font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
            font=('FixedSys', 14))
```

と

```
import random

def shuffle():
    global index, words_list, answers_list, sentences_list

    for i in range(2*len(words_list)):
        n = random.randint(0, len(words_list)-1)
        m = random.randint(0, len(words_list)-1)
        temp = words_list[n]
        words_list[n] = words_list[m]
        words_list[m] = temp
        temp = answers_list[n]
        answers_list[n] = answers_list[m]
        answers_list[m] = temp
        temp = sentences_list[n]
        sentences_list[n] = sentences_list[m]
        sentences_list[m] = temp
```

```
    index = 0
```

と定義します。これで全部のプログラミングができました。しかし、使ってみると覚えた単語と覚えられない単語が出てきます。

```
        f = open(filename, "w", encoding='utf-8')

        f = open(filename, "r", encoding='utf-8')
```

の方でプログラミングし、エディタでは、ファイルを編集し、覚えられない単語のファイルを作り、デフォルトの文字コード SHIFT-JIS ではなく、文字コードは utf-8 で保存すれば、良いと言えば良いですが、このソフトで覚えられない単語をチェックし、その単語達を保存できるようにしましょう。グローバル変数

```
checked_words_list = []
checked_answers_list = []
checked_sentences_list = []
```

を定義し、ボタン

```
button5 = Button(root, text = '覚えていない', command = check_and_show_next)
button5.pack(side=LEFT)
```

を定義し、関数 check_and_show_next() を

```
def check_and_show_next():
    global index, words_list, answers_list, sentences_list
    global checked_words_list, checked_answers_list, checked_sentences_list

    checked_words_list.append(words_list[index])
    checked_answers_list.append(answers_list[index])
    checked_sentences_list.append(sentences_list[index])

    canvas.create_rectangle(0, 0, 1000, 200, fill='white')

    index += 1
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 105, text=answers_list[index],
                font=('FixedSys', 14))
```

```
    else:
        index = 0
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
              font=('FixedSys', 14))
        else:
            canvas.create_text(500, 35, text=words_list[index],
              font=('FixedSys', 14))
```

と定義し、メニューに

```
menufile.add_command(label = "保存", under = 0, command = joseki2file)
```

を追加し、関数 joseki2file() を

```
def joseki2file():
    global M_STOP_FLAG
    global checked_words_list, checked_sentences_list, checked_answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
        ss = "[\n"
        f.write(ss)
        for ind in range(len(checked_words_list)):
            w = checked_words_list[ind]
            v = checked_answers_list[ind]
            u = checked_sentences_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(checked_words_list[len(checked_words_list)-1])
```

```
    print(checked_answers_list[len(checked_answers_list)-1])
    print(checked_sentences_list[len(checked_sentences_list)-1])
```

と定義すれば良いです。更に

```
canvas = Canvas(root, width=1000, height=200)
canvas.pack()
```

の位置をボタンの宣言の前に配置する方が良いです。
プログラムの全体は

```
from tkinter import *
import tkinter.filedialog
import sys, os.path
import random

root = Tk()

words_list = []
answers_list = []
sentences_list = []
checked_words_list = []
checked_answers_list = []
checked_sentences_list = []

path_name = ""
file_name = ""

M_STOP_FLAG = False

def joseki2file():
    global M_STOP_FLAG
    global checked_words_list, checked_sentences_list, checked_answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
        ss = "[\n"
```

```
        f.write(ss)
        for ind in range(len(checked_words_list)):
            w = checked_words_list[ind]
            v = checked_answers_list[ind]
            u = checked_sentences_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(checked_words_list[len(checked_words_list)-1])
    print(checked_answers_list[len(checked_answers_list)-1])
    print(checked_sentences_list[len(checked_sentences_list)-1])

def file2joseki():
    global M_STOP_FLAG, words_list, answers_list, sentences_list
    global checked_words_list, checked_answers_list, checked_sentences_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            answers_list.append(v)
            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
```

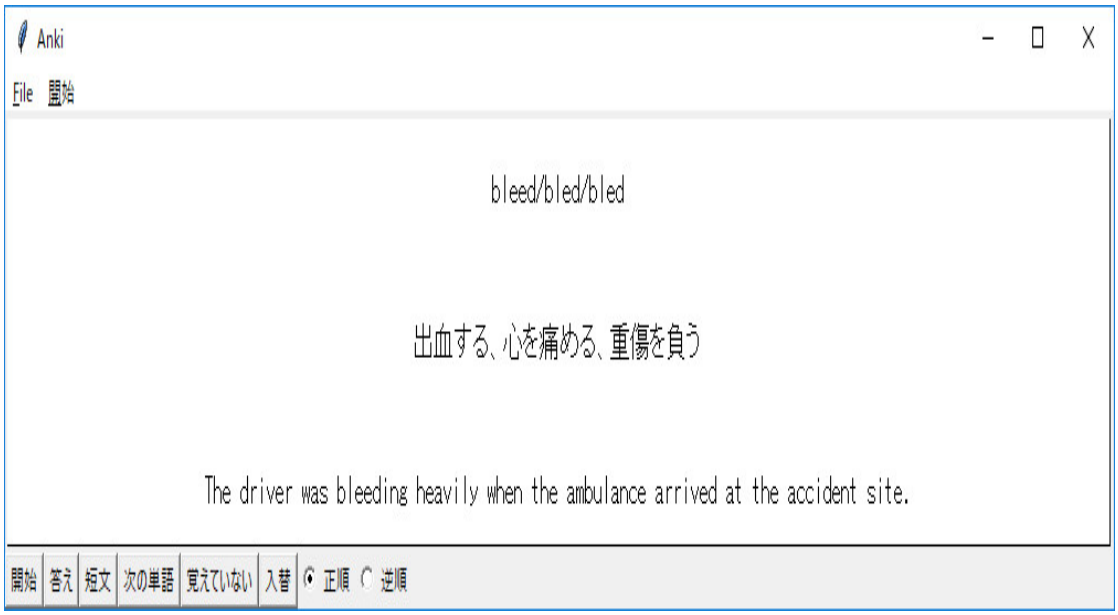
```
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
    print(sentences_list[len(sentences_list)-1])
    checked_words_list = []
    checked_answers_list = []
    checked_sentences_list = []

index = 0

def start():
    global index, words_list, answers_list, sentences_list

    canvas.create_rectangle(0, 0, 1000, 200, fill='white')

    index = 0
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 105, text=answers_list[index],
                font=('FixedSys', 14))

def show_answer():
    global index, words_list, answers_list, sentences_list

    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(500, 105, text=answers_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))

def show_sentence():
    global index, words_list, answers_list, sentences_list
```

```
    if index < len(words_list):
       canvas.create_text(500, 175, text=sentences_list[index],
           font=('FixedSys', 14))

def show_next():
    global index, words_list, answers_list, sentences_list

    canvas.create_rectangle(0, 0, 1000, 200, fill='white')

    index += 1
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 105, text=answers_list[index],
                font=('FixedSys', 14))

    else:
        index = 0
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))

def check_and_show_next():
    global index, words_list, answers_list, sentences_list
    global checked_words_list, checked_answers_list, checked_sentences_list

    checked_words_list.append(words_list[index])
    checked_answers_list.append(answers_list[index])
    checked_sentences_list.append(sentences_list[index])

    canvas.create_rectangle(0, 0, 1000, 200, fill='white')

    index += 1
    if index < len(words_list):
```

```
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 105, text=answers_list[index],
                font=('FixedSys', 14))

    else:
        index = 0
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))

def shuffle():
    global index, words_list, answers_list, sentences_list

    for i in range(2*len(words_list)):
        n = random.randint(0, len(words_list)-1)
        m = random.randint(0, len(words_list)-1)
        temp = words_list[n]
        words_list[n] = words_list[m]
        words_list[m] = temp
        temp = answers_list[n]
        answers_list[n] = answers_list[m]
        answers_list[m] = temp
        temp = sentences_list[n]
        sentences_list[n] = sentences_list[m]
        sentences_list[m] = temp
    index = 0

menubar = Menu(root)
root.configure(menu=menubar)
root.title("Anki")
val = IntVar()
val.set(0)
```

```
menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "読込", under = 0, command = file2joseki)
menufile.add_command(label = "保存", under = 0, command = joseki2file)
menubar.add_command(label = '開始', under = 0, command = start)

canvas = Canvas(root, width=1000, height=200)
canvas.pack()

button1 = Button(root, text = '開始', command = start)
button1.pack(side=LEFT)

button2 = Button(root, text = '答え', command = show_answer)
button2.pack(side=LEFT)

button5 = Button(root, text = '短文', command = show_sentence)
button5.pack(side=LEFT)

button3 = Button(root, text = '次の単語', command = show_next)
button3.pack(side=LEFT)

button5 = Button(root, text = '覚えていない', command = check_and_show_next)
button5.pack(side=LEFT)

button4 = Button(root, text = '入替', command = shuffle)
button4.pack(side=LEFT)

r0 = Radiobutton(root, text = '正順', variable = val, value = 0)
r0.pack(side=LEFT)
r1 = Radiobutton(root, text = '逆順', variable = val, value = 1)
r1.pack(side=LEFT)
##
##canvas = Canvas(root, width=1000, height=200)
##canvas.pack()
root.mainloop()
```

です。実行すると



です。間違っていないみたいです。覚えていない単語が表示されたとき、「次の単語」のボタンではなく、「覚えていない」のボタンをクリックすれば、次の単語が表示されるだけでなく、

```
checked_words_list
checked_answers_list
checked_sentences_list
```

に、覚えてない単語の情報が追加され、「File」メニューの「保存」でそれらがファイルに保存されます。

英単語とその訳だけのソフトは上のプログラムを修正して、

```
from tkinter import *
import tkinter.filedialog
import sys, os.path

root = Tk()
root.title("Anki Words Set")
canvas = Canvas(root, width = 800, height=200)
canvas.pack()



path_name = ""
file_name = ""

# 単語と意味と短文を格納する変数
```

```
word = ""
value = ""
##sentence = ""
words_list = []
answers_list = []
##sentences_list = []

M_STOP_FLAG = False

def joseki2file():
##    global M_STOP_FLAG, words_list, sentences_list, answers_list
    global M_STOP_FLAG, words_list, answers_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
        ss = "[\n"
        f.write(ss)
        for ind in range(len(words_list)):
            w = words_list[ind]
            v = answers_list[ind]
##            u = sentences_list[ind]
##            ss = "(" + w + ":" + v + ":" + u + ")\n"
            ss = "(" + w + ":" + v + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=', words_list, answers_list)

def file2joseki():
##    global M_STOP_FLAG, words_list, sentences_list, answers_list
    global M_STOP_FLAG, words_list, answers_list
```

```
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        words_list = []
        sentences_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
##            v = j_list[1]
##            u = j_list[2][:-1]
            v = j_list[1][:-1]
            words_list.append(w)
            answers_list.append(v)
##            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=', words_list, answers_list)

def start():
##    global words_list, sentences_list, answers_list
    global words_list, sentences_list

    words_list = []
    answers_list = []
##    sentences_list = []
    canvas.create_rectangle(0, 0, 800, 200, fill='green')

def add_word():
##    global word, value, sentence, words_list, sentences_list, answers_list
    global word, value, sentence, words_list, answers_list

    words_list.append(word)
```

```
    answers_list.append(value)
##    sentences_list.append(sentence)
    canvas.create_rectangle(0, 0, 800, 200, fill='white')
    buffer.set("")
    buffer2.set("")
##    buffer3.set("")

# 表示
def set_word(event):
    global word
    if buffer.get():
        word = buffer.get()
        canvas.create_text(400, 35, text=word, font=('FixedSys', 14))
        e2.focus_set()

def set_value(event):
    global value
    if buffer2.get():
        value = buffer2.get()
        canvas.create_text(400, 105, text=value, font=('FixedSys', 14))
##        e3.focus_set()
        e.focus_set()

##def set_sentence(event):
##    global sentence
##    if buffer3.get():
##        sentence = buffer3.get()
##        canvas.create_text(400, 175, text=sentence, font=('FixedSys', 14))
##        e.focus_set()

# エントリーの値を格納するオブジェクト
buffer = StringVar()
buffer.set("")
buffer2 = StringVar()
buffer2.set("")
##buffer3 = StringVar()
##buffer3.set("")
# エントリー
e = Entry(root, width=75, textvariable = buffer)
```

```
e.pack()
e.focus_set()
e2 = Entry(root, width=75, textvariable = buffer2)
e2.pack()
##e3 = Entry(root, width=75, textvariable = buffer3)
##e3.pack()
# バインディング
e.bind('<Return>', set_word)
e2.bind('<Return>', set_value)
##e3.bind('<Return>', set_sentence)

menubar = Menu(root)
root.configure(menu = menubar)
menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "保存", under = 0, command = joseki2file)
menufile.add_command(label = "読込", under = 0, command = file2joseki)

canvas.create_rectangle(0, 0, 800, 200, fill='green')
button3 = Button(root, text = '開始', command = start)
button3.pack(side=LEFT)
button1 = Button(root, text = '追加', command = add_word)
button1.pack(side=LEFT)

root.mainloop()
```

と

```
from tkinter import *
import tkinter.filedialog
import sys, os.path
import random

root = Tk()

words_list = []
answers_list = []
##sentences_list = []
checked_words_list = []
checked_answers_list = []
```

```
##checked_sentences_list = []

path_name = ""
file_name = ""

def joseki2file():
    global M_STOP_FLAG
    global checked_words_list, checked_sentences_list

    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", encoding='utf-8')
        ss = "[\n"
        f.write(ss)
        for ind in range(len(checked_words_list)):
            w = checked_words_list[ind]
            v = checked_answers_list[ind]
##            u = checked_sentences_list[ind]
##            ss = "(" + w + ":" + v + ":" + u + ")\n"
            ss = "(" + w + ":" + v + ")\n"
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(checked_words_list[len(checked_words_list)-1])
    print(checked_answers_list[len(checked_answers_list)-1])
##    print(checked_sentences_list[len(checked_sentences_list)-1])

def file2joseki():
##    global M_STOP_FLAG, words_list, answers_list, sentences_list
    global M_STOP_FLAG, words_list, answers_list

    M_STOP_FLAG = True
```

```
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r", encoding='utf-8')
        for line in f:
            line = line[:-1]
            if line.count('[')  or line.count(']'): continue
            j_list = line.split(":")
            w = j_list[0][1:]
##            v = j_list[1]
##            u = j_list[2][:-1]
            v = j_list[1][:-1]
            words_list.append(w)
            answers_list.append(v)
##            sentences_list.append(u)
        f.close()
    M_STOP_FLAG = False
##    print('words_dict=', words_list, answers_list, sentences_list)
    print('words_dict=')
    print(words_list[len(words_list)-1])
    print(answers_list[len(answers_list)-1])
##    print(sentences_list[len(sentences_list)-1])
    checked_words_list = []
    checked_answers_list = []
##    checked_sentences_list = []

index = 0

def start():
##    global index, words_list, answers_list, sentences_list
    global index, words_list, answers_list

    canvas.create_rectangle(0, 0, 800, 200, fill='white')

    index = 0
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
```

```
                font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
                font=('FixedSys', 14))

def show_answer():
##    global index, words_list, answers_list, sentences_list
    global index, words_list, answers_list

    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 105, text=answers_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(400, 35, text=words_list[index],
                font=('FixedSys', 14))

##def show_sentence():
##    global index, words_list, answers_list, sentences_list
##
##    if index < len(words_list):
##       canvas.create_text(400, 175, text=sentences_list[index], font=('FixedSys'

def show_next():
##    global index, words_list, answers_list, sentences_list
    global index, words_list, answers_list

    canvas.create_rectangle(0, 0, 800, 200, fill='white')

    index += 1
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
                font=('FixedSys', 14))

    else:
```

```
        index = 0
        if (val.get() == 0):
            canvas.create_text(400, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(400, 105, text=answers_list[index],
                font=('FixedSys', 14))

def check_and_show_next():
    global index, words_list, answers_list
    global checked_words_list, checked_answers_list

    checked_words_list.append(words_list[index])
    checked_answers_list.append(answers_list[index])
##    checked_sentences_list.append(sentences_list[index])

    canvas.create_rectangle(0, 0, 1000, 200, fill='white')

    index += 1
    if index < len(words_list):
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 105, text=answers_list[index],
                font=('FixedSys', 14))

    else:
        index = 0
        if (val.get() == 0):
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))
        else:
            canvas.create_text(500, 35, text=words_list[index],
                font=('FixedSys', 14))

def shuffle():
##    global index, words_list, answers_list, sentences_list
    global index, words_list, answers_list
```

```
    for i in range(2*len(words_list)):
        n = random.randint(0, len(words_list)-1)
        m = random.randint(0, len(words_list)-1)
        temp = words_list[n]
        words_list[n] = words_list[m]
        words_list[m] = temp
        temp = answers_list[n]
        answers_list[n] = answers_list[m]
        answers_list[m] = temp
##        temp = sentences_list[n]
##        sentences_list[n] = sentences_list[m]
##        sentences_list[m] = temp
    index = 0

menubar = Menu(root)
root.configure(menu=menubar)
root.title("Anki")

menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "読込", under = 0, command = file2joseki)
menufile.add_command(label = "保存", under = 0, command = joseki2file)
menubar.add_command(label = '開始', under = 0, command = start)

button1 = Button(root, text = '開始', command = start)
button1.pack(side=LEFT)

button2 = Button(root, text = '答え', command = show_answer)
button2.pack(side=LEFT)

##button5 = Button(root, text = '短文', command = show_sentence)
##button5.pack(side=LEFT)

button3 = Button(root, text = '次の単語', command = show_next)
button3.pack(side=LEFT)

button5 = Button(root, text = '覚えていない', command = check_and_show_next)
button5.pack(side=LEFT)
```

```
button4 = Button(root, text = '入替', command = shuffle)
button4.pack(side=LEFT)

val = IntVar()
val.set(0)
r0 = Radiobutton(root, text = '正順', variable = val, value = 0)
r0.pack(side=LEFT)
r1 = Radiobutton(root, text = '逆順', variable = val, value = 1)
r1.pack(side=LEFT)

canvas = Canvas(root, width=800, height=200)
canvas.pack()
root.mainloop()
```

です。このようにコピーアンドペーストでプログラムを修正する時には、思い込みで思わぬバグが入りこみますから、細心の注意が必要です。ファイル入出力に関する注意は、元のプログラムに関する注意を参照してください。

昔は外国語は読めさえすれば良かったのですが、最近は発音も重視され、聞くことも話すことも大事だとされています。単語の意味だけでなく、発音記号も表示するように修正してみます。文字コードに関する知識がなかったので、私が最初に作ったプログラムは次のようなものです。英文以外を使ったプログラミングでは

```
f = open(filename, "w", encoding='utf-8')

f = open(filename, "r", encoding='utf-8')
```

を使うか

```
f = open(filename, "w")

f = open(filename, "r")
```

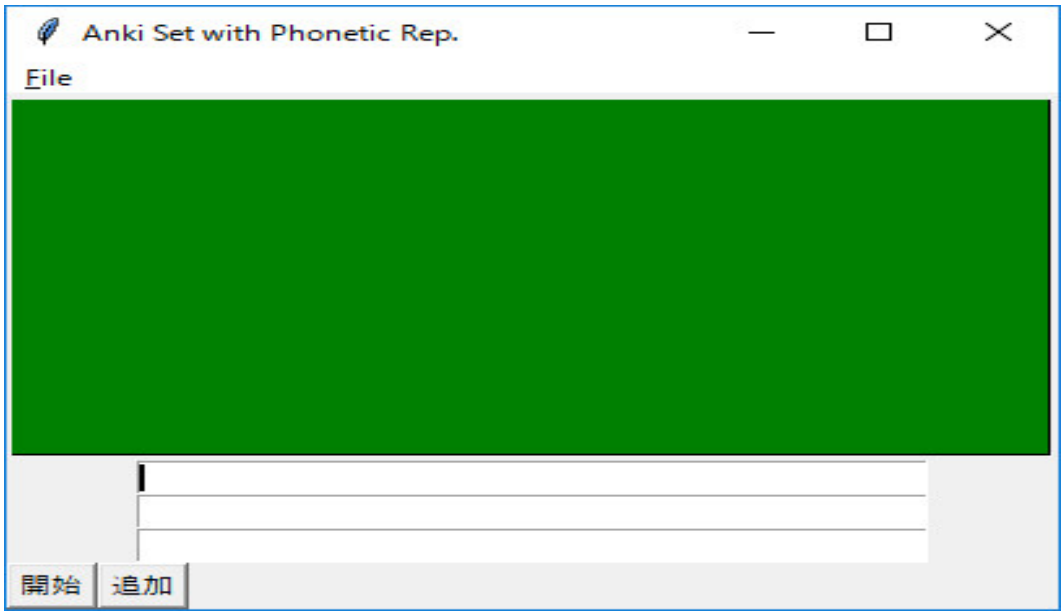
を使うかで、大きな差が出ます。

最初に作ったプログラムでは

```
f = open(filename, "w")

f = open(filename, "r")
```

を使ったので、複雑なプログラムになりました。

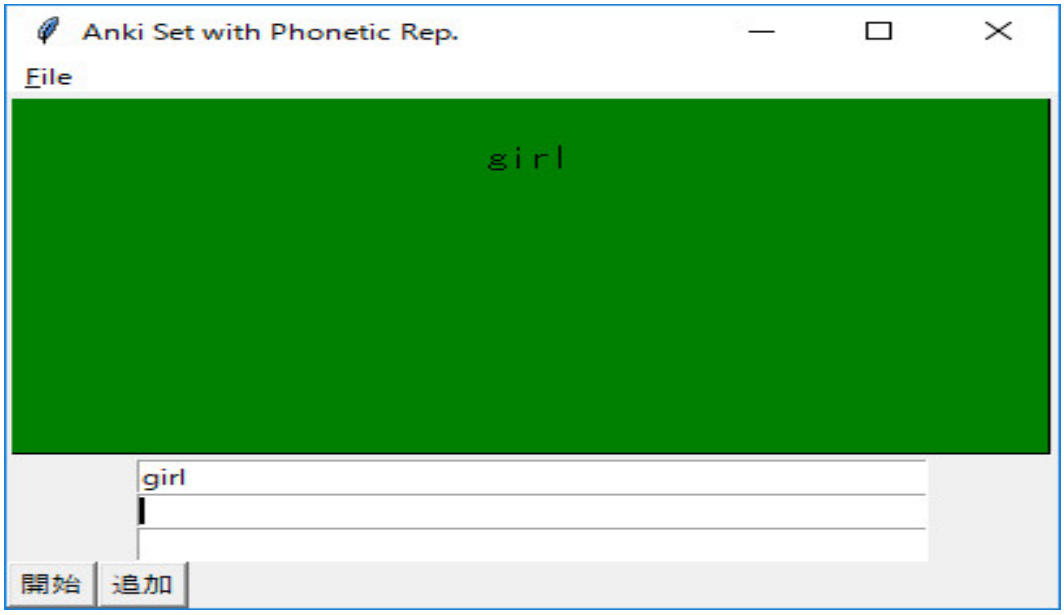
作るプログラムは２つです。まずはカードのデータを作るソフトです。立ち上げると



です。「File」メニューには「保存」と「読込」があります。「保存」は作成したデータをファイルに保存します。「読込」は作りかけのデータファイルにデータを追加するために、ファイルを読み込み、データを追加します。

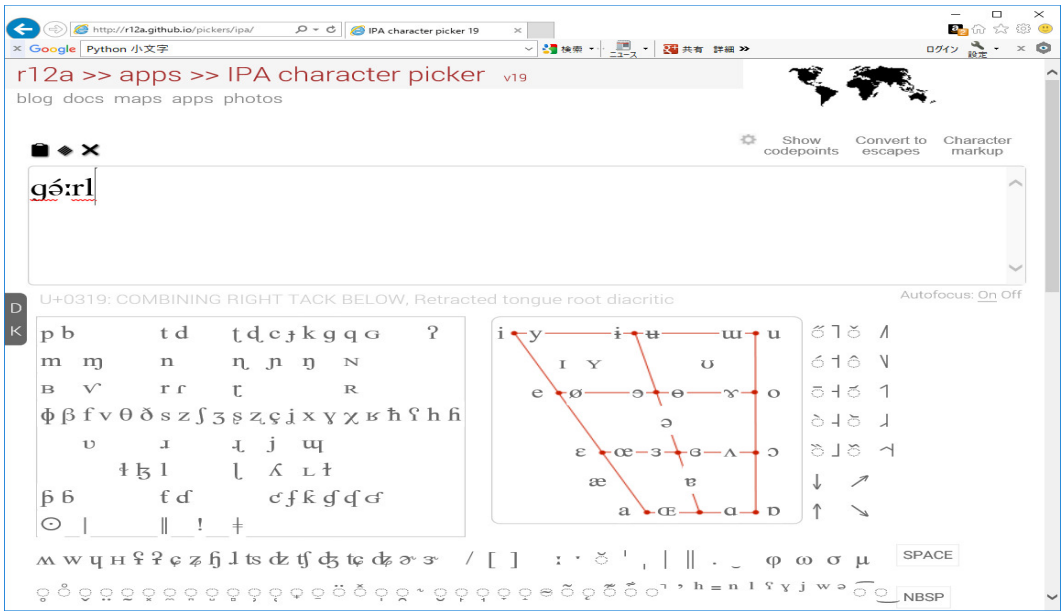
データを作成するには、「開始」のボタンをクリックします。「開始」のボタンをクリックすると辞書が初期化されます。従って、「読込」で作りかけのデータファイルを読み込んだ時には、辞書に作りかけのデータがセットされているので、「開始」のボタンをクリックすると辞書が初期化され、データの追加が出来ないので注意してください。

辞書にデータを追加するには、まず上のエントリー（テキストエディタ）に英単語（日本語でもいい）を入力し、エンターキーを押します。

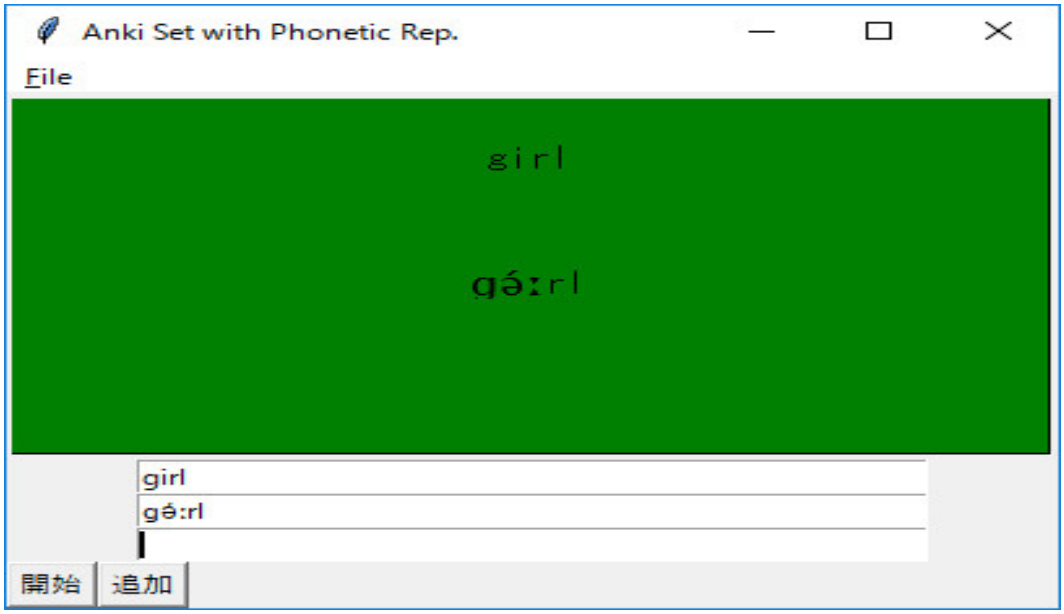


辞書のキーがセットされました。次に、真ん中のエントリー（テキストエディタ）に発音記号を入力します。

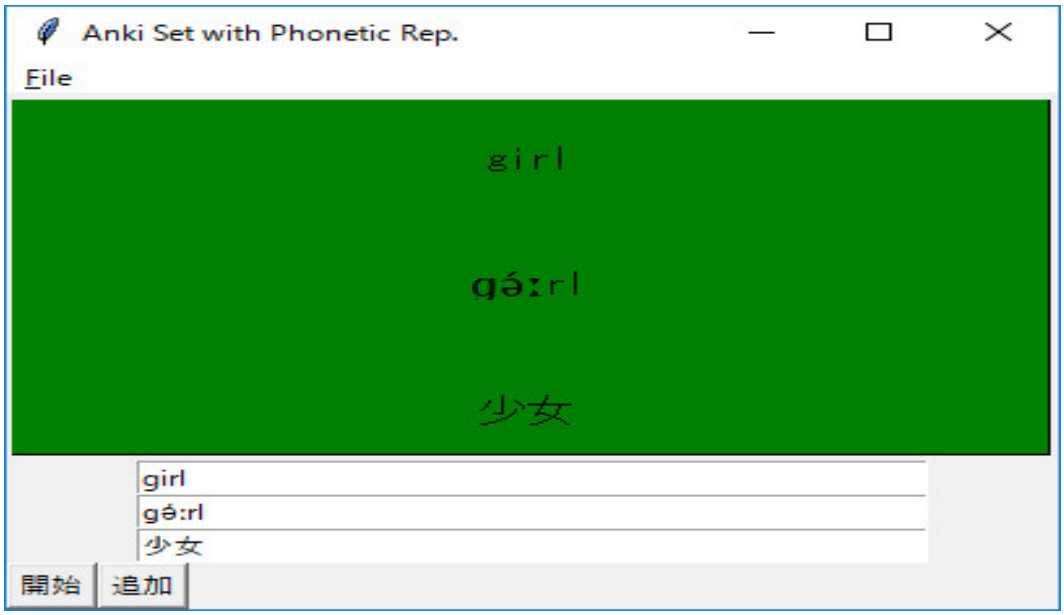
この為には、例えば、http://r12a.github.io/pickers/ipa/ を開き、



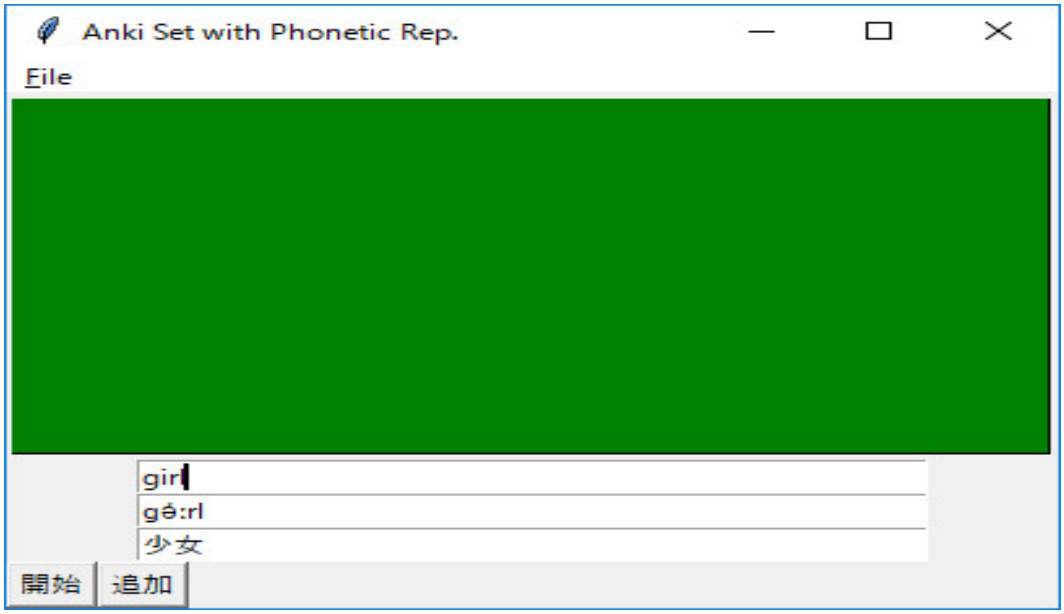
発音記号の文字列を作り、発音記号をコピーし、エンターキーを押します。



次に、下のエントリー（テキストエディタ）に日本語（英単語でもいい）を入力し、エンターキーを押します。



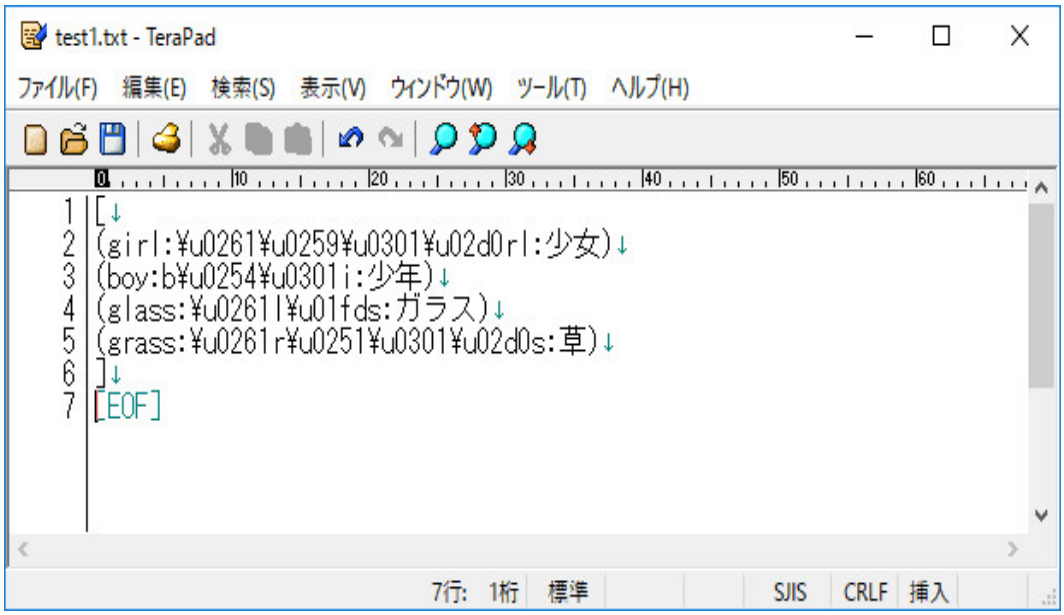
辞書の値がセットされました。これで、辞書のキーと値が揃ったので、実際に辞書にそのデータを追加するために、「追加」のボタンをクリックします。



本当はエントリー（テキストエディタ）達を空にしておくべきでした。必要なら自分で修正してください。

この手順を繰り返して、単語カードのデータを作ります。データが出来たら、ファイルに保存します。「File」メニューの「保存」を使って保存します。

保存したファイルは

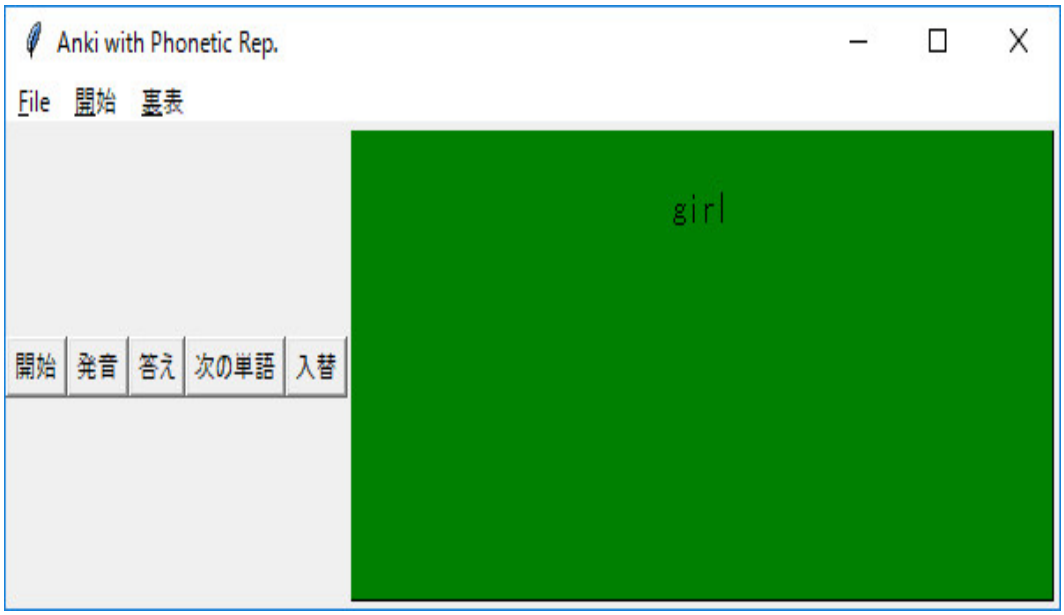


のようなテキストファイルです。従って、全く同じ形式でデータを作ることによって、テキストエディターで、単語カードを編集することが出来ます。最初の行は [だけ、最後の行は] だけ、辞書の中身は (キーの単語:発音記号:値の単語) です。[ ] (:) はすべて半角です。

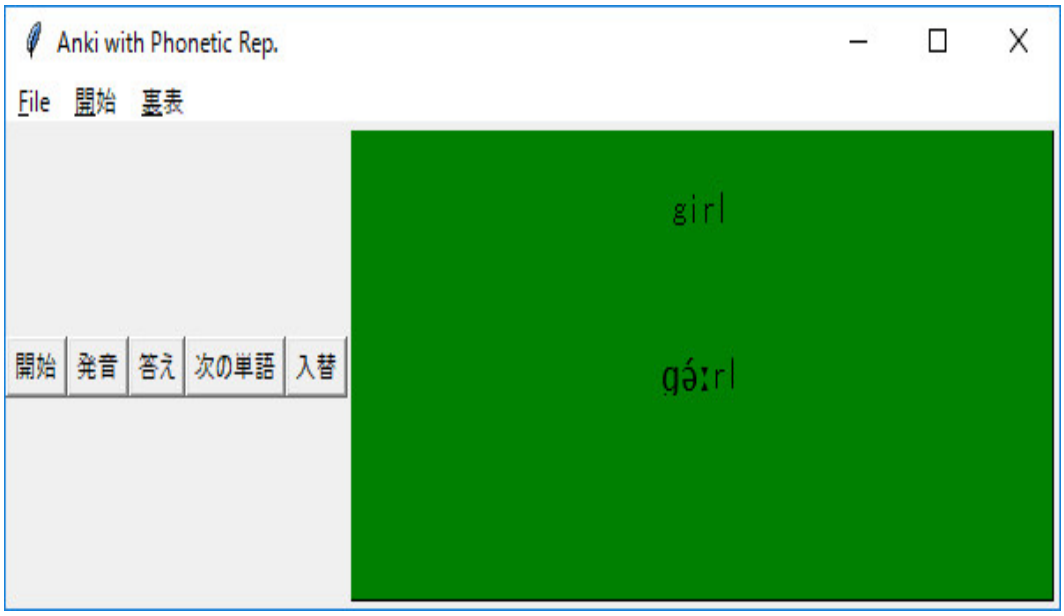
次は、ここで作成したファイルを読み込んで、順番に表示するソフトです。立ち上げると

です。「File」メニューには「読込」があります。「読込」で保存してあるファイルを読み込みます。先ほどのファイルを読み込んでみます。

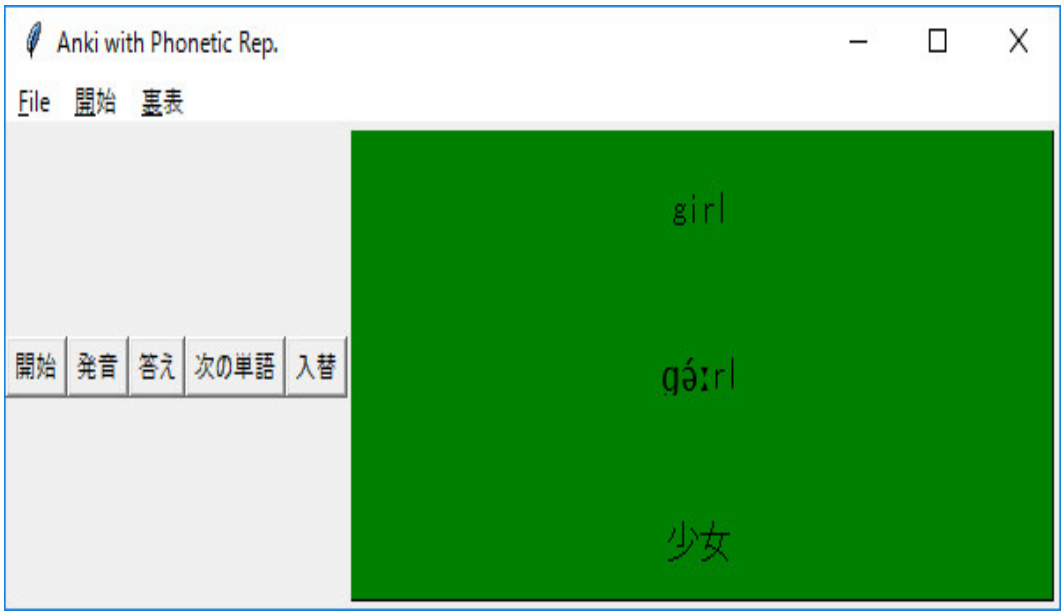
「開始」のボタンかメニューの開始をクリックします。



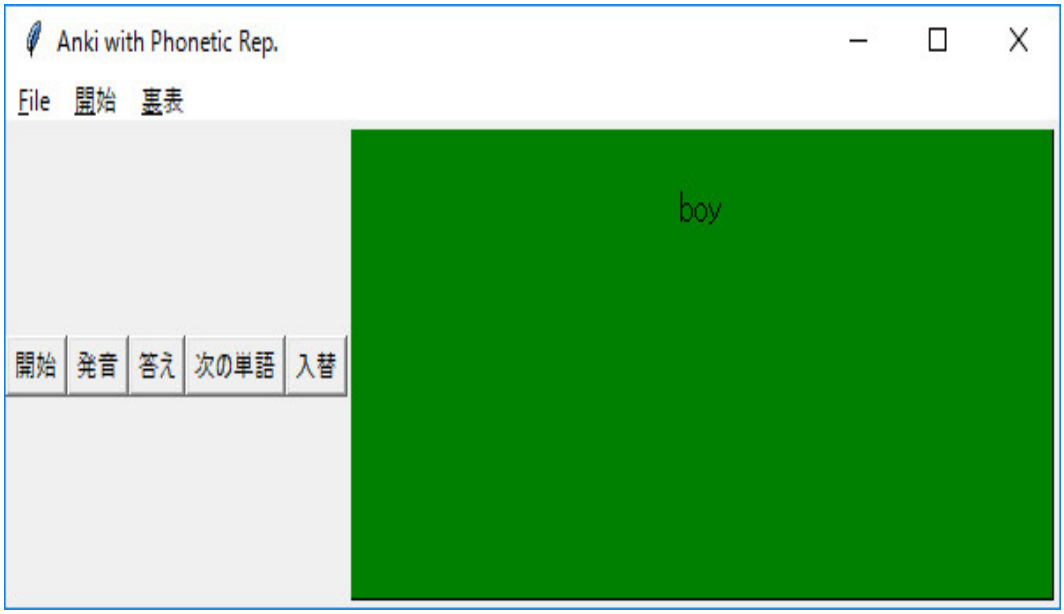
最初の単語が表示されます。「発音」のボタンを押すと



キーと結び付けられている値が表示されます。「答え」のボタンを押すと



キーと結び付けられている値が表示されます。「次の単語」のボタンを押すと



次のキーが表示されます。

「裏表」のメニューには「表」と「裏」のサブメニューがあり、デフォルトでは「表」がチェックされています。「裏」をチェックすると、「答え」のボタンを押すとキーが表示され、「開始」及び「次の単語」のボタンを押すとキーと結び付けられている値が表示されます。単語カードを裏返ししたことに当たります。

「入替」のボタンを押すと辞書がランダムにシャッフルされます。

プログラムリストは

```
from tkinter import *
```

```
import tkinter.filedialog
import sys, os.path

root = Tk() ##root.option_add("*font", ("FixedSys", 10))
root.title("Anki Set with Phonetic Rep.")

phonetic_code = { 'u0070':u'\u0070', 'u0062':u'\u0062','u0074':u'\u0074',
                  'u0064':u'\u0064','u0288':u'\u0288', 'u0256':u'\u0256',
                  'u0063':u'\u0063','u025F':u'\u025F', 'u006B':u'\u006B',
                  'u0261':u'\u0261','u0071':u'\u0071', 'u0262':u'\u0262',
                  'u0294':u'\u0294','u006D':u'\u006D', 'u0271':u'\u0271',
                  'u006e':u'\u006e','u0273':u'\u0273', 'u0272':u'\u0272',
                  'u014b':u'\u014b','u0274':u'\u0274', 'u0299':u'\u0299',
                  'u0072':u'\u0072','u0280':u'\u0280', 'u027e':u'\u027e',
                  'u027d':u'\u027d','u0278':u'\u0278', 'u03b2':u'\u03b2',
                  'u0066':u'\u0066','u0076':u'\u0076', 'u03b8':u'\u03b8',
                  'u00f0':u'\u00f0','u0073':u'\u0073', 'u007a':u'\u007a',
                  'u0283':u'\u0283','u0292':u'\u0292', 'u0282':u'\u0282',
                  'u0290':u'\u0290','u00e7':u'\u00e7', 'u029d':u'\u029d',
                  'u0078':u'\u0078','u0263':u'\u0263', 'u03c7':u'\u03c7',
                  'u0281':u'\u0281','u0127':u'\u0127', 'u0295':u'\u0295',
                  'u0068':u'\u0068','u0266':u'\u0266', 'u026c':u'\u026c',
                  'u026e':u'\u026e','u028b':u'\u028b', 'u0279':u'\u0279',
                  'u027b':u'\u027b','u006a':u'\u006a', 'u0270':u'\u0270',
                  'u006c':u'\u006c','u026d':u'\u026d', 'u028e':u'\u028e',
                  'u029f':u'\u029f','u01a5':u'\u01a5', 'u0253':u'\u0253',
                  'u01ad':u'\u01ad','u0257':u'\u0257', 'u0188':u'\u0188',
                  'u0284':u'\u0284','u0199':u'\u0199', 'u0260':u'\u0260',
                  'u02a0':u'\u02a0','u029b':u'\u029b', 'u028d':u'\u028d',
                  'u0077':u'\u0077','u0265':u'\u0265', 'u029c':u'\u029c',
                  'u02a1':u'\u02a1','u02a2':u'\u02a2', 'u0267':u'\u0267',
                  'u0298':u'\u0298','u01c0':u'\u01c0', 'u01c3':u'\u01c3',
                  'u01c2':u'\u01c2','u01c1':u'\u01c1', 'u027a':u'\u027a',
                  'u0255':u'\u0255','u0291':u'\u0291', 'u2c71':u'\u2c71',
                  'u0287':u'\u0287','u0297':u'\u0297', 'u0296':u'\u0296',
                  'u0286':u'\u0286','u0293':u'\u0293', 'u027c':u'\u027c',
                  'u02e2':u'\u02e2','u01ab':u'\u01ab', 'u026b':u'\u026b',
                  'u0067':u'\u0067','u02a6':u'\u02a6', 'u02a3':u'\u02a3',
                  'u02a7':u'\u02a7','u02a4':u'\u02a4', 'u02a8':u'\u02a8',
```

```
'u02a5':u'\u02a5','u1dbf':u'\u1dbf', 'u1d4a':u'\u1d4a',
'u1d91':u'\u1d91','u01bb':u'\u01bb', 'u029e':u'\u029e',
'u02e3':u'\u02e3','u019e':u'\u019e', 'u019b':u'\u019b',
'u03bb':u'\u03bb','u017e':u'\u017e', 'u0161':u'\u0161',
'u01f0':u'\u01f0','u010d':u'\u010d', 'u0069':u'\u0069',
'u0065':u'\u0065','u025b':u'\u025b', 'u0061':u'\u0061',
'u0251':u'\u0251','u0254':u'\u0254', 'u006f':u'\u006f',
'u0075':u'\u0075','u0079':u'\u0079', 'u00f8':u'\u00f8',
'u0153':u'\u0153','u0276':u'\u0276', 'u0252':u'\u0252',
'u028c':u'\u028c','u0264':u'\u0264', 'u026f':u'\u026f',
'u0268':u'\u0268','u0289':u'\u0289', 'u026a':u'\u026a',
'u028f':u'\u028f','u028a':u'\u028a', 'u0259':u'\u0259',
'u0275':u'\u0275','u0250':u'\u0250', 'u00e6':u'\u00e6',
'u025c':u'\u025c','u025a':u'\u025a', 'u0131':u'\u0131',
'u025e':u'\u025e','u029a':u'\u029a', 'u0258':u'\u0258',
'u0277':u'\u0277','u0269':u'\u0269', 'u02bc':u'\u02bc',
'u0325':u'\u0325','u030a':u'\u030a', 'u032c':u'\u032c',
'u02b0':u'\u02b0','u0324':u'\u0324', 'u0330':u'\u0330',
'u033c':u'\u033c','u032a':u'\u032a', 'u033a':u'\u033a',
'u033b':u'\u033b','u0339':u'\u0339', 'u031c':u'\u031c',
'u031f':u'\u031f','u0320':u'\u0320', 'u0308':u'\u0308',
'u033d':u'\u033d','u0318':u'\u0318', 'u0319':u'\u0319',
'u02de':u'\u02de','u02b7':u'\u02b7', 'u02b2':u'\u02b2',
'u02e0':u'\u02e0','u02e4':u'\u02e4', 'u0303':u'\u0303',
'u207f':u'\u207f','u02e1':u'\u02e1', 'u031a':u'\u031a',
'u0334':u'\u0334','u031d':u'\u031d', 'u02d4':u'\u02d4',
'u031e':u'\u031e','u02d5':u'\u02d5', 'u0329':u'\u0329',
'u032f':u'\u032f','u0361':u'\u0361', 'u0322':u'\u0322',
'u002c':u'\u002c','u02bb':u'\u02bb', 'u0307':u'\u0307',
'u02d7':u'\u02d7','u02d6':u'\u02d6', 'u02b8':u'\u02b8',
'u0323':u'\u0323','u0321':u'\u0321', 'u032b':u'\u032b',
'u02c8':u'\u02c8','u02cc':u'\u02cc', 'u02d0':u'\u02d0',
'u02d1':u'\u02d1','u0306':u'\u0306', 'u002e':u'\u002e',
'u007c':u'\u007c','u2016':u'\u2016', 'u203f':u'\u203f',
'u2197':u'\u2197','u2198':u'\u2198', 'u030b':u'\u030b',
'u0301':u'\u0301','u0304':u'\u0304', 'u0300':u'\u0300',
'u030f':u'\u030f','u2193':u'\u2193', 'u2191':u'\u2191',
'u02e5':u'\u02e5','u02e6':u'\u02e6', 'u02e7':u'\u02e7',
'u02e8':u'\u02e8','u02e9':u'\u02e9', 'u030c':u'\u030c',
```

```
                    'u0302':u'\u0302','u1dc4':u'\u1dc4', 'u1dc5':u'\u1dc5',
                    'u1dc8':u'\u1dc8','u0311':u'\u0311', 'u02c7':u'\u02c7',
                    'u02c6':u'\u02c6','u02ce':u'\u02ce', 'u02cf':u'\u02cf',
                    'u005b':u'\u005b','u005d':u'\u005d', 'u002f':u'\u002f',
                    'u0028':u'\u0028','u0029':u'\u0029', 'u007b':u'\u007b',
                    'u007d':u'\u007d','u01fd':u'\u01fd'
                   }

path_name = ""
file_name = ""

words_list = []
phonetics_list = []
answers_list = []

M_STOP_FLAG = False

def joseki2file():
    global M_STOP_FLAG
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.asksaveasfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "w", errors='backslashreplace')
        ss = "[\n"
        f.write(ss)
        for ind in range(len(words_list)):
            w = words_list[ind]
            v = phonetics_list[ind]
            u = answers_list[ind]
            ss = "(" + w + ":" + v + ":" + u + ")\n"
            print(ss)
            f.write(ss)
        ss = "]\n"
        f.write(ss)
        f.close()
    M_STOP_FLAG = False
```

```
def start():
    global words_list, phonetics_list, answers_list

    words_list = []
    phonetics_list = []
    answers_list = []

    canvas.create_rectangle(0, 0, 400, 200, fill='green')

def add_word():
    global word, symbol, value, words_list, phonetics_list, answers_list

    words_list.append(word)
    phonetics_list.append(symbol)
    answers_list.append(value)
    canvas.create_rectangle(0, 0, 400, 200, fill='green')

# 単語と発音と意味を格納するオブジェクト
word = ""
symbol = ""
value = ""

# 式を格納するオブジェクト
buffer = StringVar()
buffer.set("")
buffer2 = StringVar()
buffer2.set("")
buffer3 = StringVar()
buffer3.set("")

# 計算
def set_word(event):
    global word
    if buffer.get():
        word = buffer.get()
        canvas.create_text(200, 35, text=word, font=('FixedSys', 14))
        e2.focus_set()

def set_symbol(event):
```

```
    global symbol
    if buffer2.get():
        symbol = buffer2.get()
        canvas.create_text(200, 105, text=symbol, font=('FixedSys', 14))
        e3.focus_set()

def set_value(event):
    global value
    if buffer3.get():
        value = buffer3.get()
        canvas.create_text(200, 175, text=value, font=('FixedSys', 14))
        e.focus_set()

def unicode2str(v):
    result = ""
    u_list = v.split('\\')
    for u in u_list:
        if len(u) > 0 and u[0] == 'u':
            if len(u) == 5:
                if u in phonetic_code:
                    result += phonetic_code[u]
                else:
                    result += phonetic_code[u.lower()]
            elif len(u) > 5:
                if u[0:5] in phonetic_code:
                    result += phonetic_code[u[0:5]]
                else:
                    result += phonetic_code[u[0:5].lower()]
                for i in range(5,len(u)):
                    result += u[i]
        else:
            result += u
    return result

def file2joseki():
    global M_STOP_FLAG, words_list, phonetics_list, answers_list
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
```

```
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r")
        words_list = []
        phonetics_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
            if line == '['  or line == ']': continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
            phonetics_list.append(unicode2str(v))
            answers_list.append(u)

        f.close()
    M_STOP_FLAG = False
    print('words_dict=', words_list, phonetics_list, answers_list)



menubar = Menu(root)
root.configure(menu = menubar)
menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "保存", under = 0, command = joseki2file)
menufile.add_command(label = "読込", under = 0, command = file2joseki)

canvas = Canvas(root, width = 400, height=200)
canvas.pack()

# エントリー
e = Entry(root, width=50, textvariable = buffer)
e.pack()
e.focus_set()
e2 = Entry(root, width=50, textvariable = buffer2)
e2.pack()
```

```
e3 = Entry(root, width=50, textvariable = buffer3)
e3.pack()

# バインディング
e.bind('<Return>', set_word)
e2.bind('<Return>', set_symbol)
e3.bind('<Return>', set_value)

canvas.create_rectangle(0, 0, 400, 200, fill='green')
button3 = Button(root, text = '開始', command = start)
button3.pack(side=LEFT)
button1 = Button(root, text = '追加', command = add_word)
button1.pack(side=LEFT)

root.mainloop()
```

と

```
from tkinter import *
import tkinter.filedialog
import sys, os.path
import random


root = Tk()

words_list = ["girl", "boy", "glass", "grass"]
phonetics_list = ["\u0261\u0259\u0301\u02d0rl",
                  "b\u0254\u0301i",
                  "\u0261l\u01fds",
                  "\u0261r\u0251\u0301\u02d0s"]
answers_list = ["少女", "少年", "グラス", "草"]

phonetic_code = { 'u0070':u'\u0070', 'u0062':u'\u0062','u0074':u'\u0074',
                  'u0064':u'\u0064','u0288':u'\u0288', 'u0256':u'\u0256',
                  'u0063':u'\u0063','u025F':u'\u025F', 'u006B':u'\u006B',
                  'u0261':u'\u0261','u0071':u'\u0071', 'u0262':u'\u0262',
                  'u0294':u'\u0294','u006D':u'\u006D', 'u0271':u'\u0271',
                  'u006e':u'\u006e','u0273':u'\u0273', 'u0272':u'\u0272',
                  'u014b':u'\u014b','u0274':u'\u0274', 'u0299':u'\u0299',
```

```
’u0072’:u’\u0072’,’u0280’:u’\u0280’, ’u027e’:u’\u027e’,
’u027d’:u’\u027d’,’u0278’:u’\u0278’, ’u03b2’:u’\u03b2’,
’u0066’:u’\u0066’,’u0076’:u’\u0076’, ’u03b8’:u’\u03b8’,
’u00f0’:u’\u00f0’,’u0073’:u’\u0073’, ’u007a’:u’\u007a’,
’u0283’:u’\u0283’,’u0292’:u’\u0292’, ’u0282’:u’\u0282’,
’u0290’:u’\u0290’,’u00e7’:u’\u00e7’, ’u029d’:u’\u029d’,
’u0078’:u’\u0078’,’u0263’:u’\u0263’, ’u03c7’:u’\u03c7’,
’u0281’:u’\u0281’,’u0127’:u’\u0127’, ’u0295’:u’\u0295’,
’u0068’:u’\u0068’,’u0266’:u’\u0266’, ’u026c’:u’\u026c’,
’u026e’:u’\u026e’,’u028b’:u’\u028b’, ’u0279’:u’\u0279’,
’u027b’:u’\u027b’,’u006a’:u’\u006a’, ’u0270’:u’\u0270’,
’u006c’:u’\u006c’,’u026d’:u’\u026d’, ’u028e’:u’\u028e’,
’u029f’:u’\u029f’,’u01a5’:u’\u01a5’, ’u0253’:u’\u0253’,
’u01ad’:u’\u01ad’,’u0257’:u’\u0257’, ’u0188’:u’\u0188’,
’u0284’:u’\u0284’,’u0199’:u’\u0199’, ’u0260’:u’\u0260’,
’u02a0’:u’\u02a0’,’u029b’:u’\u029b’, ’u028d’:u’\u028d’,
’u0077’:u’\u0077’,’u0265’:u’\u0265’, ’u029c’:u’\u029c’,
’u02a1’:u’\u02a1’,’u02a2’:u’\u02a2’, ’u0267’:u’\u0267’,
’u0298’:u’\u0298’,’u01c0’:u’\u01c0’, ’u01c3’:u’\u01c3’,
’u01c2’:u’\u01c2’,’u01c1’:u’\u01c1’, ’u027a’:u’\u027a’,
’u0255’:u’\u0255’,’u0291’:u’\u0291’, ’u2c71’:u’\u2c71’,
’u0287’:u’\u0287’,’u0297’:u’\u0297’, ’u0296’:u’\u0296’,
’u0286’:u’\u0286’,’u0293’:u’\u0293’, ’u027c’:u’\u027c’,
’u02e2’:u’\u02e2’,’u01ab’:u’\u01ab’, ’u026b’:u’\u026b’,
’u0067’:u’\u0067’,’u02a6’:u’\u02a6’, ’u02a3’:u’\u02a3’,
’u02a7’:u’\u02a7’,’u02a4’:u’\u02a4’, ’u02a8’:u’\u02a8’,
’u02a5’:u’\u02a5’,’u1dbf’:u’\u1dbf’, ’u1d4a’:u’\u1d4a’,
’u1d91’:u’\u1d91’,’u01bb’:u’\u01bb’, ’u029e’:u’\u029e’,
’u02e3’:u’\u02e3’,’u019e’:u’\u019e’, ’u019b’:u’\u019b’,
’u03bb’:u’\u03bb’,’u017e’:u’\u017e’, ’u0161’:u’\u0161’,
’u01f0’:u’\u01f0’,’u010d’:u’\u010d’, ’u0069’:u’\u0069’,
’u0065’:u’\u0065’,’u025b’:u’\u025b’, ’u0061’:u’\u0061’,
’u0251’:u’\u0251’,’u0254’:u’\u0254’, ’u006f’:u’\u006f’,
’u0075’:u’\u0075’,’u0079’:u’\u0079’, ’u00f8’:u’\u00f8’,
’u0153’:u’\u0153’,’u0276’:u’\u0276’, ’u0252’:u’\u0252’,
’u028c’:u’\u028c’,’u0264’:u’\u0264’, ’u026f’:u’\u026f’,
’u0268’:u’\u0268’,’u0289’:u’\u0289’, ’u026a’:u’\u026a’,
’u028f’:u’\u028f’,’u028a’:u’\u028a’, ’u0259’:u’\u0259’,
’u0275’:u’\u0275’,’u0250’:u’\u0250’, ’u00e6’:u’\u00e6’,
```

```
                    'u025c':u'\u025c','u025a':u'\u025a', 'u0131':u'\u0131',
                    'u025e':u'\u025e','u029a':u'\u029a', 'u0258':u'\u0258',
                    'u0277':u'\u0277','u0269':u'\u0269', 'u02bc':u'\u02bc',
                    'u0325':u'\u0325','u030a':u'\u030a', 'u032c':u'\u032c',
                    'u02b0':u'\u02b0','u0324':u'\u0324', 'u0330':u'\u0330',
                    'u033c':u'\u033c','u032a':u'\u032a', 'u033a':u'\u033a',
                    'u033b':u'\u033b','u0339':u'\u0339', 'u031c':u'\u031c',
                    'u031f':u'\u031f','u0320':u'\u0320', 'u0308':u'\u0308',
                    'u033d':u'\u033d','u0318':u'\u0318', 'u0319':u'\u0319',
                    'u02de':u'\u02de','u02b7':u'\u02b7', 'u02b2':u'\u02b2',
                    'u02e0':u'\u02e0','u02e4':u'\u02e4', 'u0303':u'\u0303',
                    'u207f':u'\u207f','u02e1':u'\u02e1', 'u031a':u'\u031a',
                    'u0334':u'\u0334','u031d':u'\u031d', 'u02d4':u'\u02d4',
                    'u031e':u'\u031e','u02d5':u'\u02d5', 'u0329':u'\u0329',
                    'u032f':u'\u032f','u0361':u'\u0361', 'u0322':u'\u0322',
                    'u002c':u'\u002c','u02bb':u'\u02bb', 'u0307':u'\u0307',
                    'u02d7':u'\u02d7','u02d6':u'\u02d6', 'u02b8':u'\u02b8',
                    'u0323':u'\u0323','u0321':u'\u0321', 'u032b':u'\u032b',
                    'u02c8':u'\u02c8','u02cc':u'\u02cc', 'u02d0':u'\u02d0',
                    'u02d1':u'\u02d1','u0306':u'\u0306', 'u002e':u'\u002e',
                    'u007c':u'\u007c','u2016':u'\u2016', 'u203f':u'\u203f',
                    'u2197':u'\u2197','u2198':u'\u2198', 'u030b':u'\u030b',
                    'u0301':u'\u0301','u0304':u'\u0304', 'u0300':u'\u0300',
                    'u030f':u'\u030f','u2193':u'\u2193', 'u2191':u'\u2191',
                    'u02e5':u'\u02e5','u02e6':u'\u02e6', 'u02e7':u'\u02e7',
                    'u02e8':u'\u02e8','u02e9':u'\u02e9', 'u030c':u'\u030c',
                    'u0302':u'\u0302','u1dc4':u'\u1dc4', 'u1dc5':u'\u1dc5',
                    'u1dc8':u'\u1dc8','u0311':u'\u0311', 'u02c7':u'\u02c7',
                    'u02c6':u'\u02c6','u02ce':u'\u02ce', 'u02cf':u'\u02cf',
                    'u005b':u'\u005b','u005d':u'\u005d', 'u002f':u'\u002f',
                    'u0028':u'\u0028','u0029':u'\u0029', 'u007b':u'\u007b',
                    'u007d':u'\u007d','u01fd':u'\u01fd'
                   }

path_name = ""
file_name = ""

def unicode2str(v):
    result = ""
```

```
    u_list = v.split('\\')
    for u in u_list:
        if len(u) > 0 and u[0] == 'u':
            if len(u) == 5:
                if u in phonetic_code:
                    result += phonetic_code[u]
                else:
                    result += phonetic_code[u.lower()]
            elif len(u) > 5:
                if u[0:5] in phonetic_code:
                    result += phonetic_code[u[0:5]]
                else:
                    result += phonetic_code[u[0:5].lower()]
                for i in range(5,len(u)):
                    result += u[i]
        else:
            result += u
    return result


def file2joseki():
    global M_STOP_FLAG, words_list, phonetics_list, answers_list
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r")
        words_list = []
        phonetics_list = []
        answers_list = []
        for line in f:
            line = line[:-1]
            if line == '['  or line == ']': continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1]
            u = j_list[2][:-1]
            words_list.append(w)
```

```
            phonetics_list.append(unicode2str(v))
            answers_list.append(u)

        f.close()
    M_STOP_FLAG = False
    print('words_dict=', words_list, phonetics_list, answers_list)

index = 0

def start():
    global index, words_list, phonetics_list, answers_list

    canvas.create_rectangle(0, 0, 400, 200, fill='green')

    index = 0
    if index < len(words_list):
        if teban.get() == 1:
            canvas.create_text(200, 35, text=words_list[index],
                                font=('FixedSys', 14))
        else:
            canvas.create_text(200, 35, text=answers_list[index],
                                font=('FixedSys', 14))


def show_phonetic():
    global index, words_list, phonetics_list, answers_list

    if index < len(words_list):
       canvas.create_text(200, 105, text=phonetics_list[index],
                            font=('FixedSys', 14))

def show_answer():
    global index, words_list, phonetics_list, answers_list

    if index < len(words_list):
        if teban.get() == 1:
            canvas.create_text(200, 175, text=answers_list[index],
                                font=('FixedSys', 14))
        else:
```

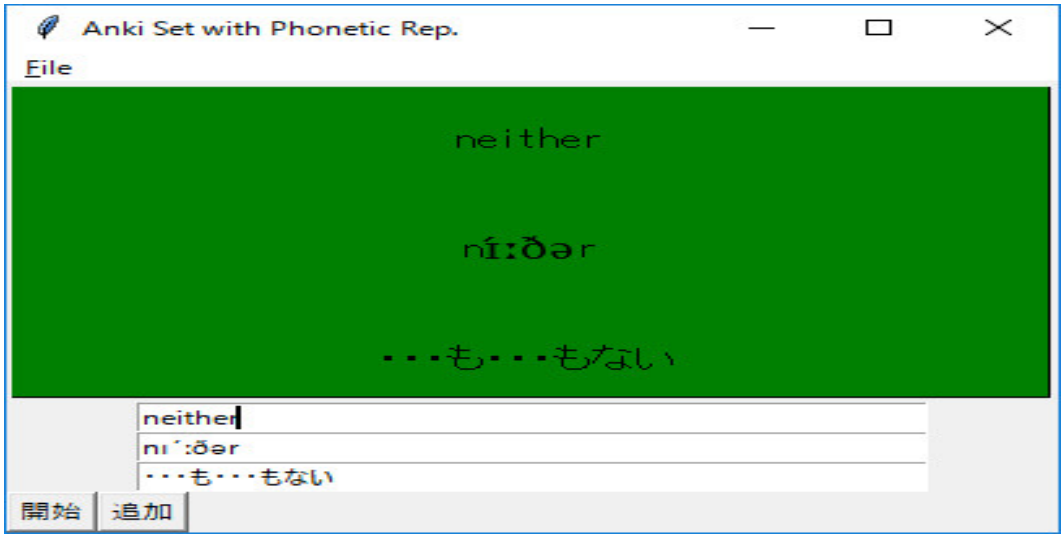
```
            canvas.create_text(200, 175, text=words_list[index],
                                    font=('FixedSys', 14))


def show_next():
    global index, words_list, phonetics_list, answers_list

    canvas.create_rectangle(0, 0, 400, 200, fill='green')

    index += 1
    if index < len(words_list):
        if teban.get() == 1:
            canvas.create_text(200, 35, text=words_list[index],
                                    font=('FixedSys', 14))
        else:
            canvas.create_text(200, 35, text=answers_list[index],
                                    font=('FixedSys', 14))
    else:
        index = 0
        if teban.get() == 1:
            canvas.create_text(200, 35, text=words_list[index],
                                    font=('FixedSys', 14))
        else:
            canvas.create_text(200, 35, text=answers_list[index],
                                    font=('FixedSys', 14))

def shuffle():
    global index, words_list, phonetics_list, answers_list

    for i in range(2*len(words_list)):
        n = random.randint(0, len(words_list)-1)
        m = random.randint(0, len(words_list)-1)
        temp = words_list[n]
        words_list[n] = words_list[m]
        words_list[m] = temp
        temp = phonetics_list[n]
        phonetics_list[n] = phonetics_list[m]
        phonetics_list[m] = temp
        temp = answers_list[n]
```

```
        answers_list[n] = answers_list[m]
        answers_list[m] = temp
    index = 0

# variable 用のオブジェクト
teban = IntVar()
teban.set(1)

menubar = Menu(root)
root.configure(menu=menubar)
root.title("Anki with Phonetic Rep.")

menufile = Menu(menubar, tearoff = False)
menubar.add_cascade(label="File", underline = 0, menu=menufile)
menufile.add_command(label = "読込", under = 0, command = file2joseki)
menubar.add_command(label = '開始', under = 0, command = start)
tebans = Menu(menubar, tearoff = False)
menubar.add_cascade(label="裏表", underline = 0, menu=tebans)
tebans.add_radiobutton(label = '表', variable = teban, value = 1)
tebans.add_radiobutton(label = '裏', variable = teban, value = 2)

button1 = Button(root, text = '開始', command = start)
button1.pack(side=LEFT)

button5 = Button(root, text = '発音', command = show_phonetic)
button5.pack(side=LEFT)

button2 = Button(root, text = '答え', command = show_answer)
button2.pack(side=LEFT)

button3 = Button(root, text = '次の単語', command = show_next)
button3.pack(side=LEFT)

button4 = Button(root, text = '入替', command = shuffle)
button4.pack(side=LEFT)

canvas = Canvas(root, width=400, height=200)
canvas.pack()
root.mainloop()
```

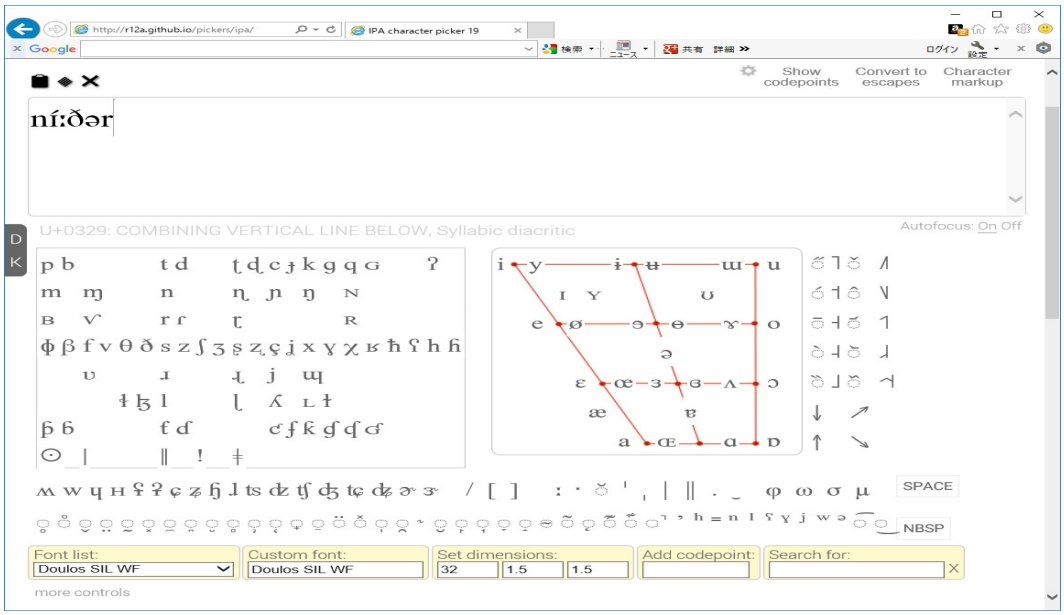
です。

誰でも作れる単純なプログラムですが、私は気に入っています。お役に立つならご自由にお使いください。画面の色や画面の大きさや文字のサイズが気に入らなければ、自分で変えてください。Unicode には、同じ文字で複数のコードを持っているのがあります。プログラムの辞書に登録してない文字コードがあって、エラーが出れば、何が登録されてないかエラー表示を見れば分かりますから、`phonetic_code` に自分で追加してください。やり方はプログラムコードを見れば分かります。単純な変換です。ファイルに保存される発音記号の Unicode を Python で使える Unicode に変換する方法をプログラミングで簡単に行う方法がどうしてもわからなかったので、辞書形式で切り抜けました。文字コードには色々あって厄介です。私は上のようなプログラミングしか思いつきませんでしたが、もっと簡単な方法があるかもしれません。

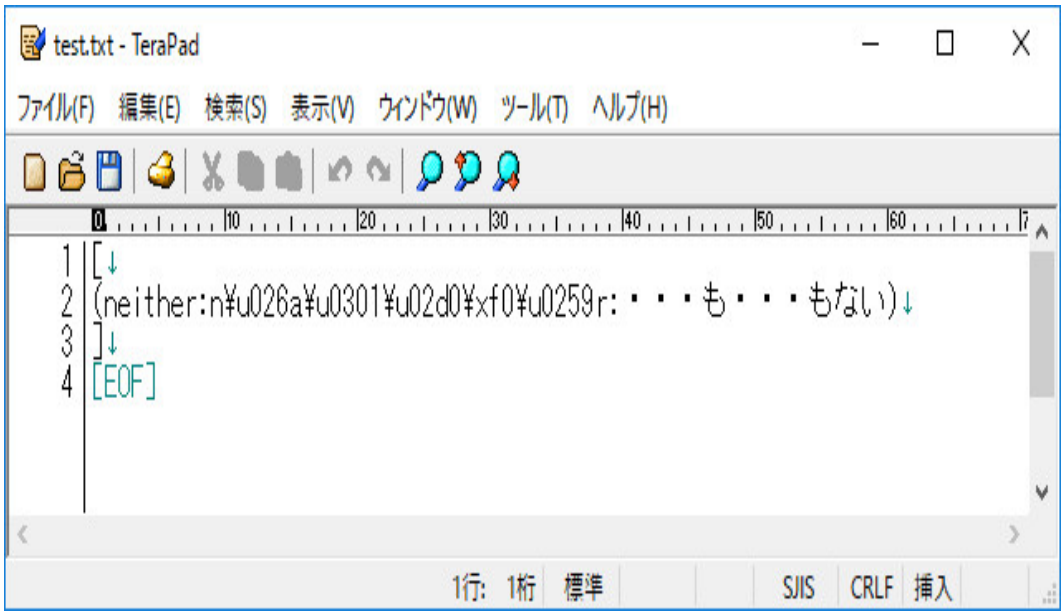
所で



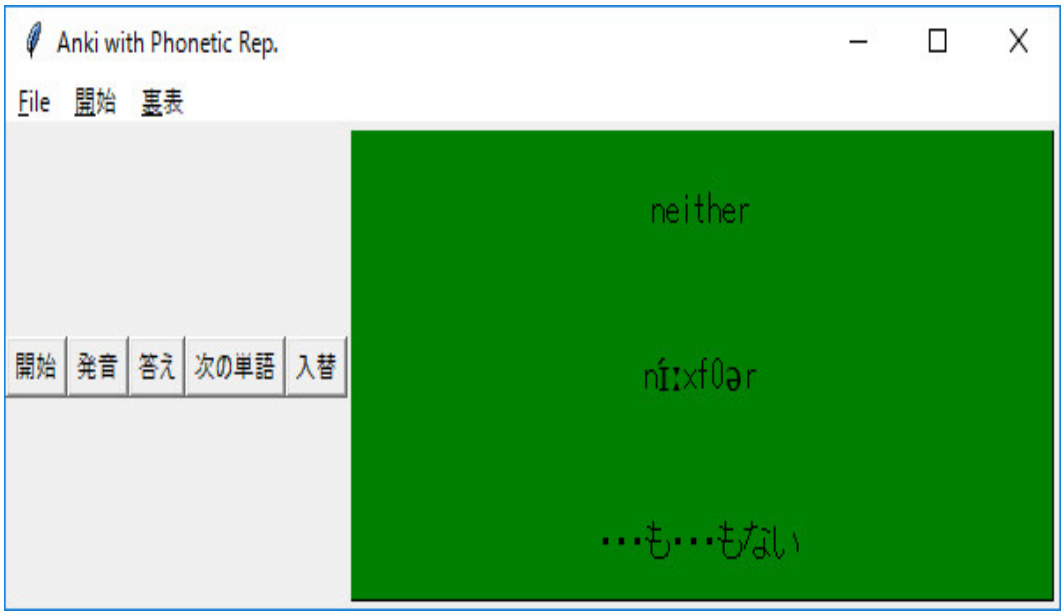
と



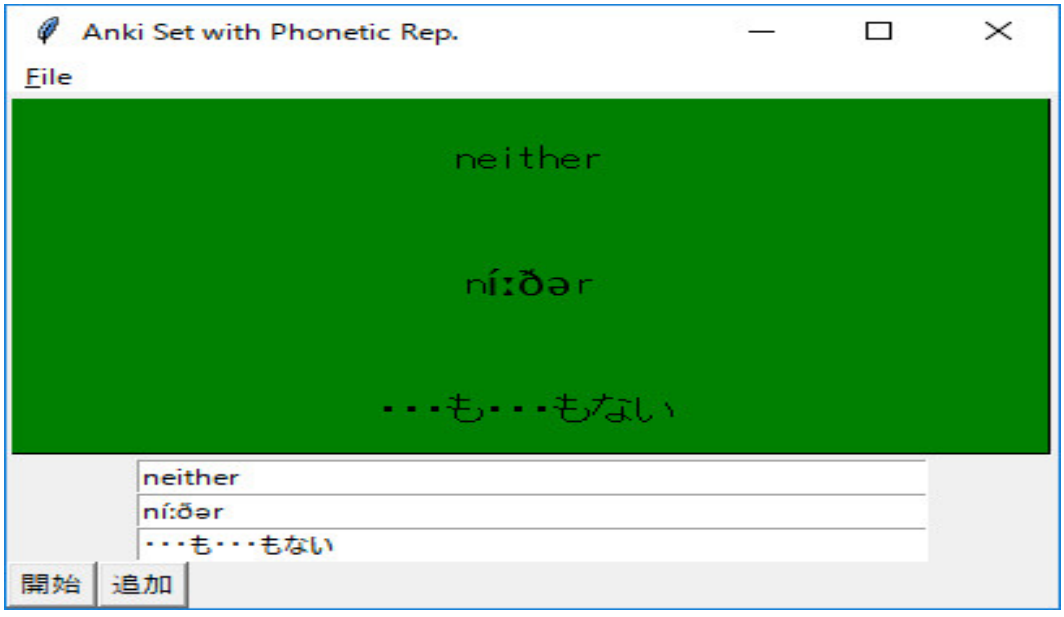
を使って発音記号を入力した時、ファイルに保存すると



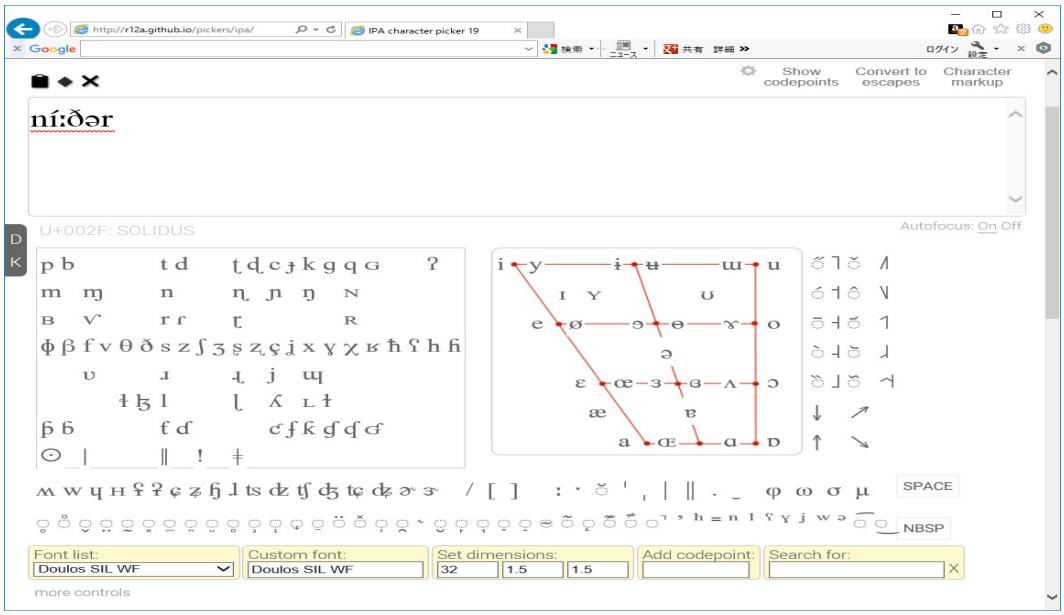
となります。th の発音記号が `\xf0` と変換されています。このファイルを読み込むと



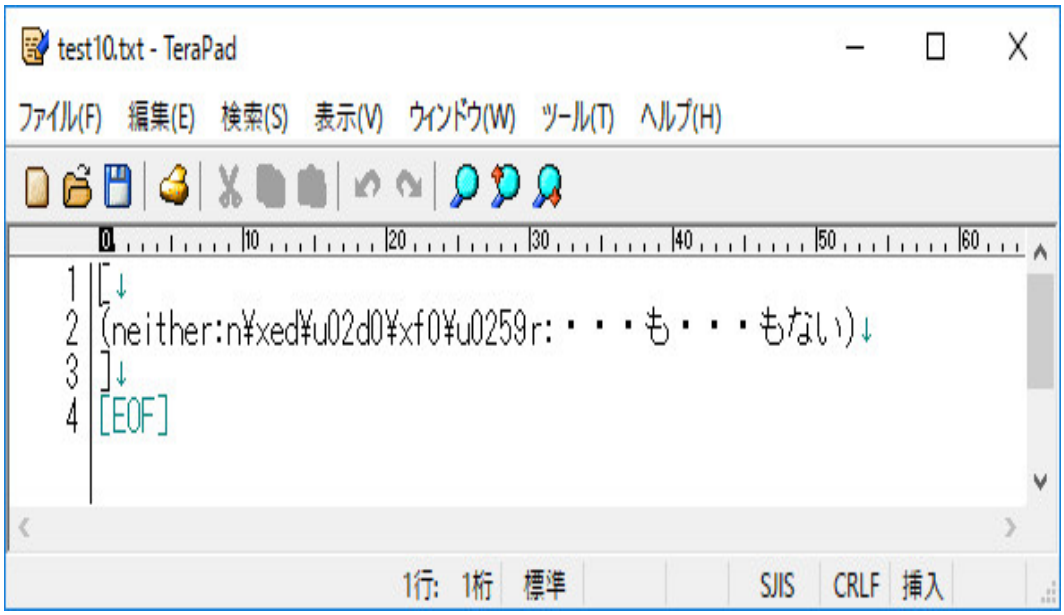
の表示になります。気になるなら単純な方法はファイルをエディターで\xf0 を \u00f0 と手で変換してください。ファイルに 発音記号を utf-8 ？で保存する方法が分からないのですが th の発音記号だけはなぜだか utf-8 ？で保存されています。UCS 符号で１６進２桁で表せるものと４桁で表せるものの保存方法が今の私のプログラムでは保存方法が別々みたいです。更に



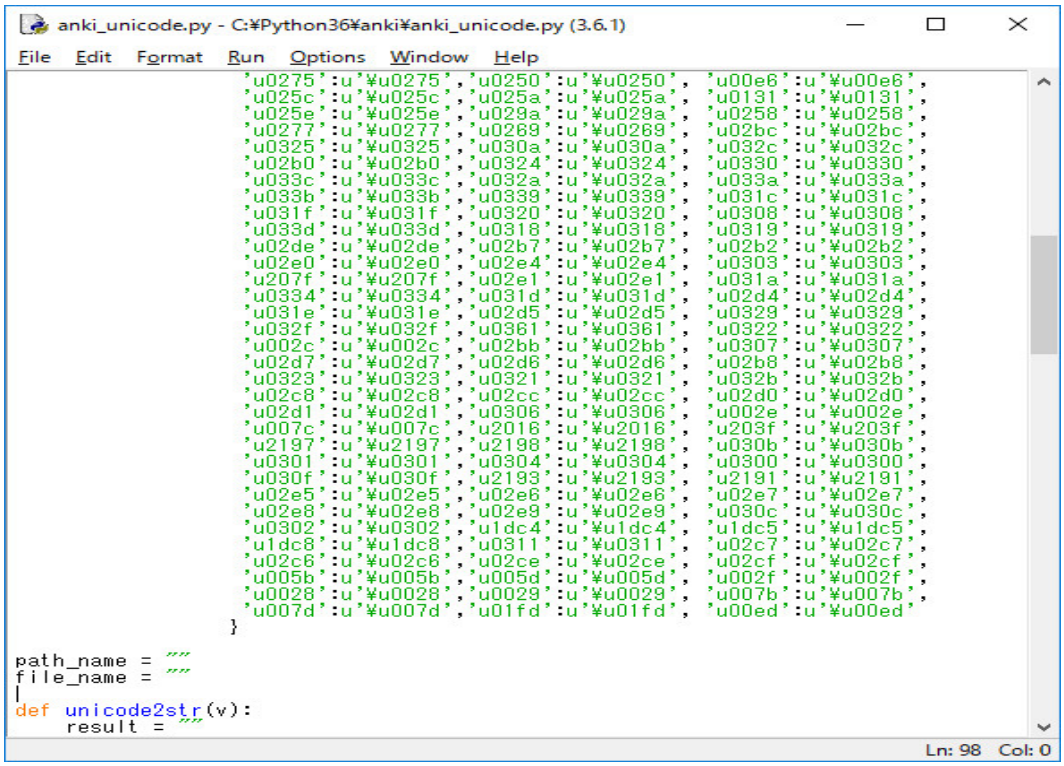
と



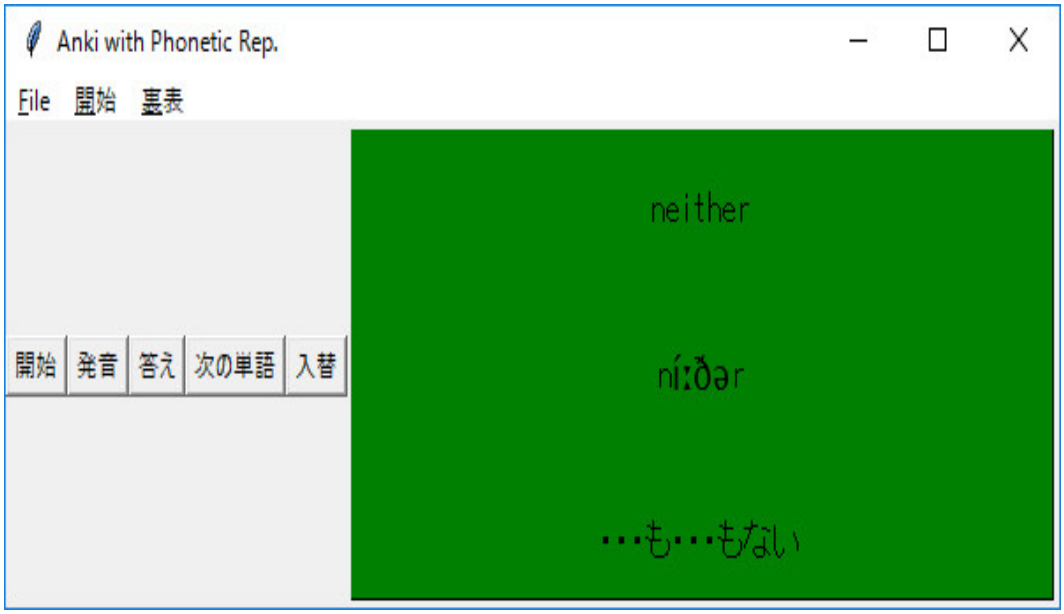
を使って i を使って発音記号を入力した時、ファイルに保存すると



となります。i の長音が \xed と変換されています。ファイルをエディターで\xf0 を \u00f0 と \xed を \u00ed と手で変換し、更にプログラムを



と辞書の最後に，'u00ed':u'\u00ed' を追加すれば、



と表示してくれるようになります。他にもバグがあるかも分かりません。バグが見つかるたびに修正しなくていけなくて、なかなか厄介です。ファイルが絡んでくると Unicode の処理はまだまだ Python では私には難しいです。ファイルをいじるよりプログラムを修正して対応するほうがいいですが、最初に示したプログラムに比べ、このプログラムはあまり使わないと思いますので、私が修正したものの全リストは示しません。二か所ある

```
def unicode2str(v):
    result = ""
    u_list = v.split('\\')
    for u in u_list:
        if len(u) > 0 and u[0] == 'u':
            if len(u) == 5:
                if u in phonetic_code:
                    result += phonetic_code[u]
                else:
                    result += phonetic_code[u.lower()]
            elif len(u) > 5:
                if u[0:5] in phonetic_code:
                    result += phonetic_code[u[0:5]]
                else:
                    result += phonetic_code[u[0:5].lower()]
                for i in range(5,len(u)):
                    result += u[i]
        else:
            result += u
    return result
```

を

```
def unicode2str(v):
    result = ""
    u_list = v.split('\\')
    for u in u_list:
        if len(u) > 0 and u[0] == 'u':
            if len(u) == 5:
                if u in phonetic_code:
                    result += phonetic_code[u]
                else:
                    result += phonetic_code[u.lower()]
            elif len(u) > 5:
                if u[0:5] in phonetic_code:
                    result += phonetic_code[u[0:5]]
                else:
                    result += phonetic_code[u[0:5].lower()]
                for i in range(5,len(u)):
```

```
                result += u[i]
        elif  len(u) > 0 and u[0] == 'x':
            u = 'u00'+u[1:]
            if len(u) == 5:
                if u in phonetic_code:
                    result += phonetic_code[u]
                else:
                    result += phonetic_code[u.lower()]
            elif len(u) > 5:
                if u[0:5] in phonetic_code:
                    result += phonetic_code[u[0:5]]
                else:
                    result += phonetic_code[u[0:5].lower()]
                for i in range(5,len(u)):
                    result += u[i]
        else:
            result += u
    return result
```

と修正すれば良いので、必要なら自分で修正してください。

```
f = open(filename, "w")
```

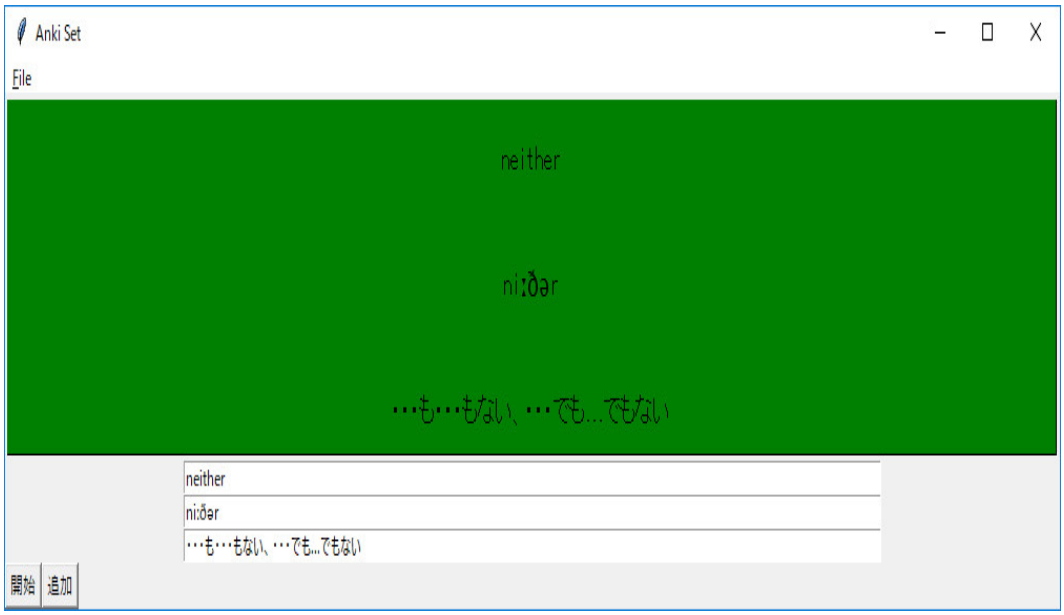
```
f = open(filename, "r")
```

を使うとこのようにめんどくさいです。しかし、

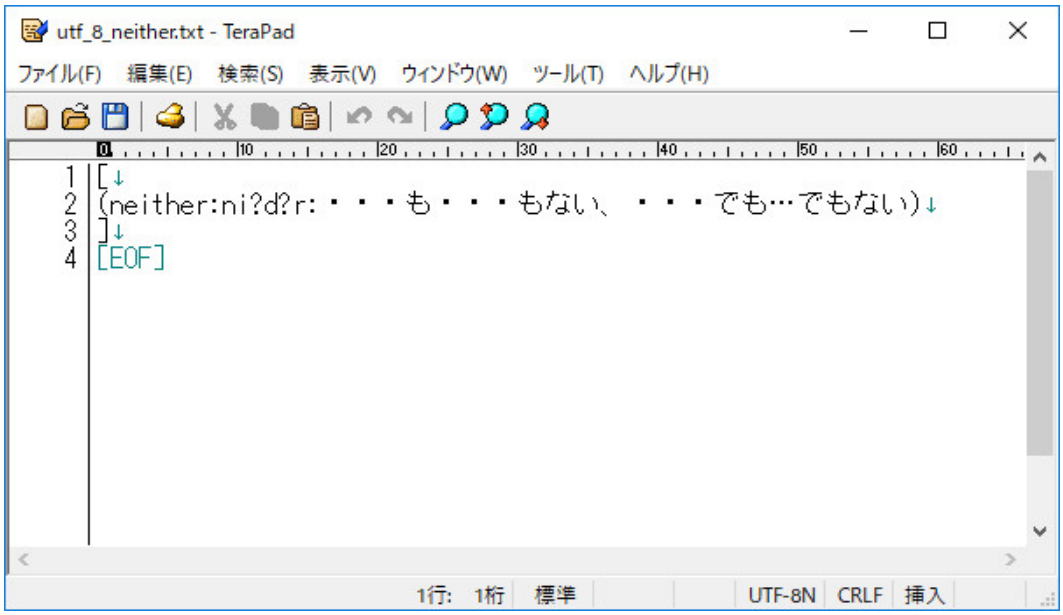
```
f = open(filename, "w", encoding='utf-8')
```

```
f = open(filename, "r", encoding='utf-8')
```

を使うとこの場合、上で作ったプログラムがそのまま使えます。



を追加し、保存すると TeraPad では

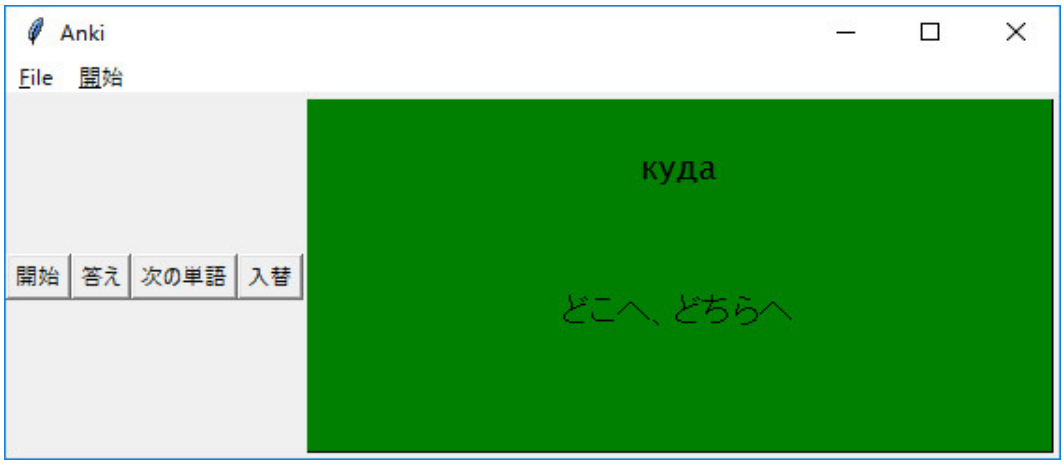


ですが、上で作ったソフトで読み込むと

と表示してくれます。

ロシア語を使いたい場合は、「設定」の「時刻と言語」の「地域と言語」の「言語を追加する」で「ロシア語」を Windows 10 に追加し、言語をロシア語に変更し、キーボードを

の読み替えで、入力すれば



のように問題なく使えます。ロシア語を使う場合は

```
f = open(filename, "w")
```

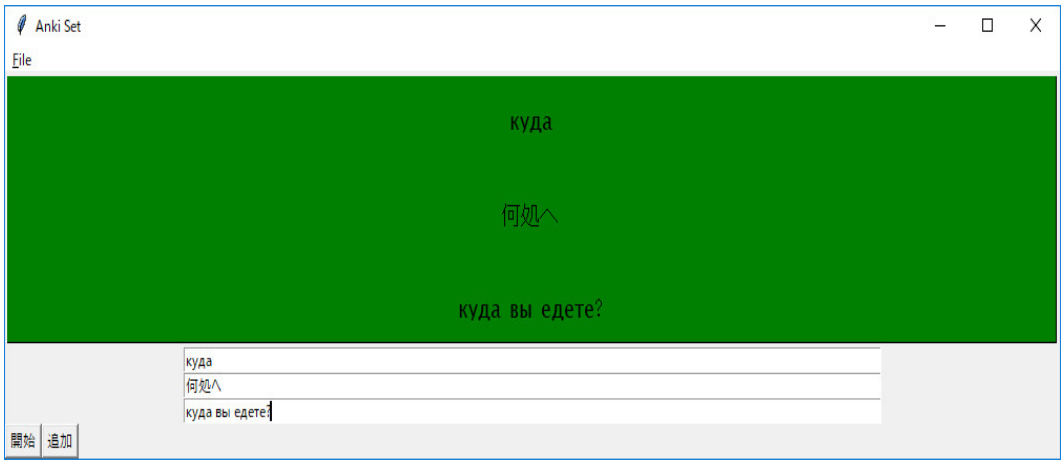
```
f = open(filename, "r")
```

でも

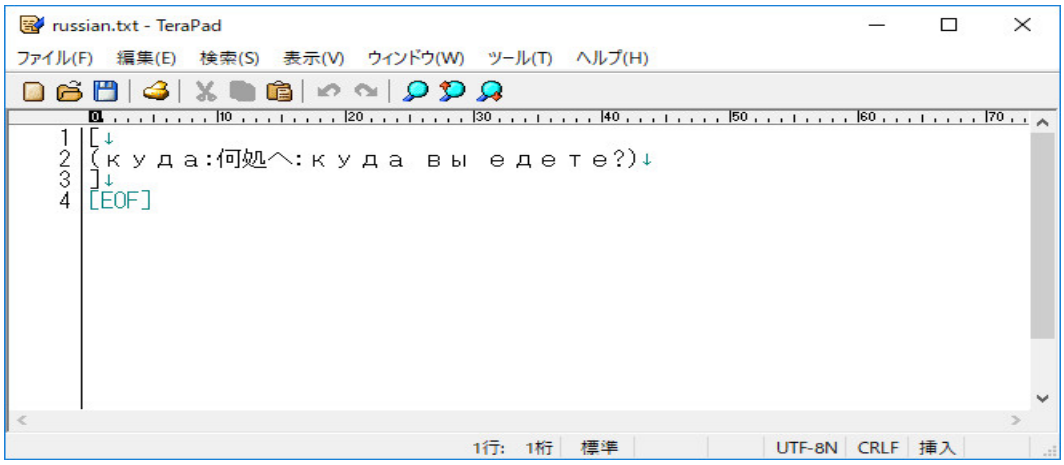
```
f = open(filename, "w", encoding='utf-8')
```

```
f = open(filename, "r", encoding='utf-8')
```

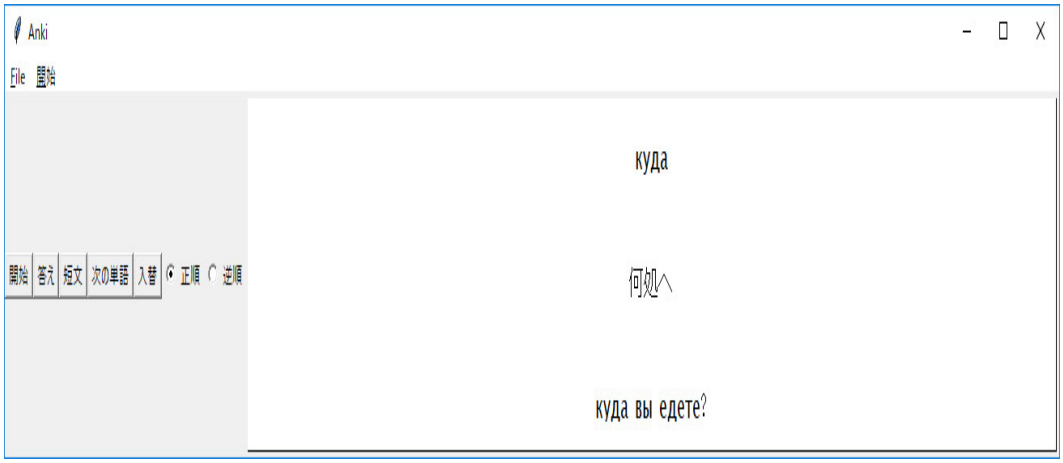
でも良いです。上で作ったプログラムで



と入力し、保存すると



となり、上で作ったプログラムで

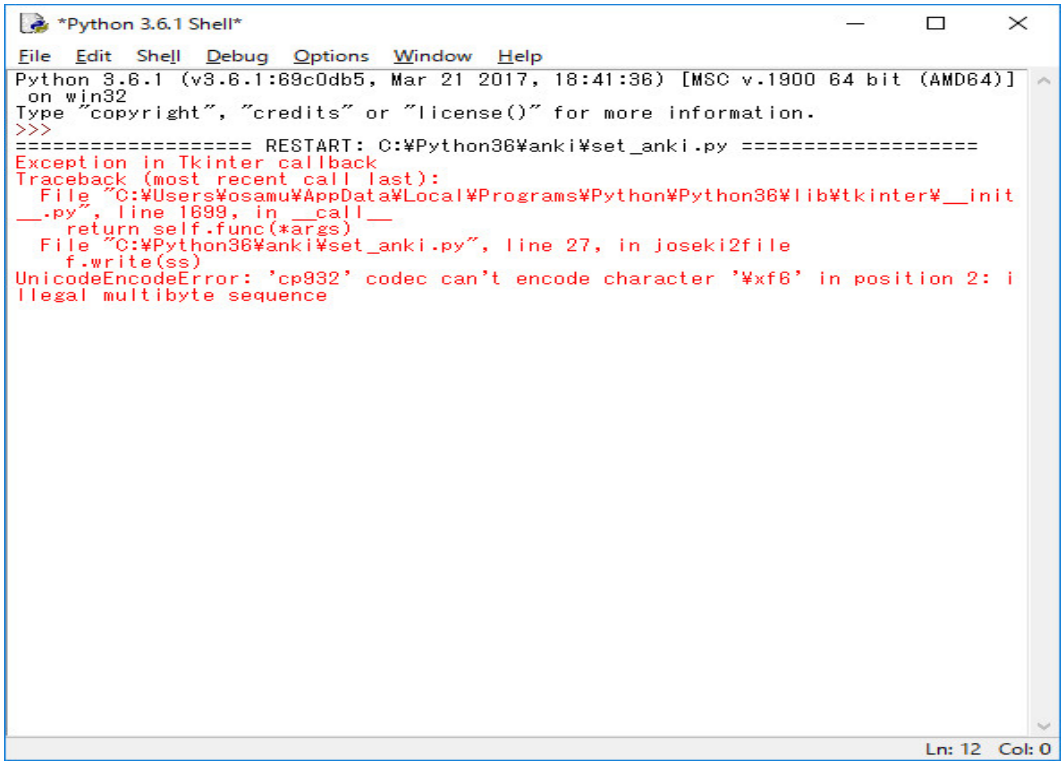


のように、問題なく使えます。
　ドイツ語の場合は、同様にして、

```
f = open(filename, "w")

f = open(filename, "r")
```

の場合は

と打ち込んで、保存すると



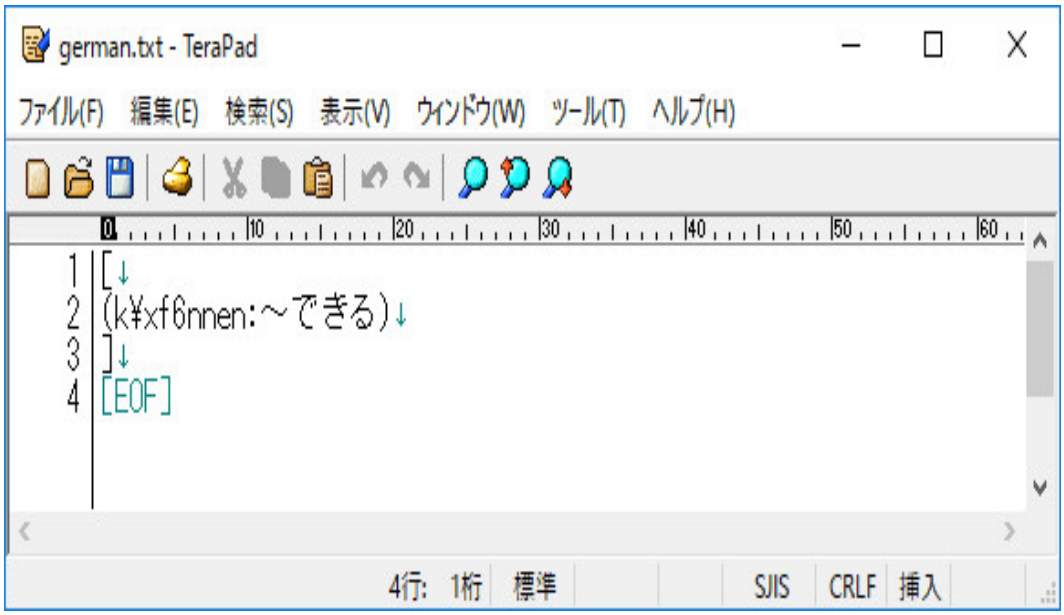
とエラーメッセージが出て、保存してくれません。発音記号を使う２番目のプログラムのように

```
f = open(filename, "w")
```

を

```
f = open(filename, "w", errors='backslashreplace')
```

と変えるとファイルに書き込んでくれますが、ファイルは



となっています。Anki を実行し、このファイルを読み込むと

となります。ドイツ語を使いたければ、発音記号に対して行ったと同じ処理が必要になります。つまり、

```
def file2joseki():
    global M_STOP_FLAG, words_dict
    M_STOP_FLAG = True
    global path_name, joseki
    filename = tkinter.filedialog.askopenfilename(initialdir=path_name)
    if filename:
        path_name = os.path.dirname(filename)
        f = open(filename, "r")
        words_dict = {}
```

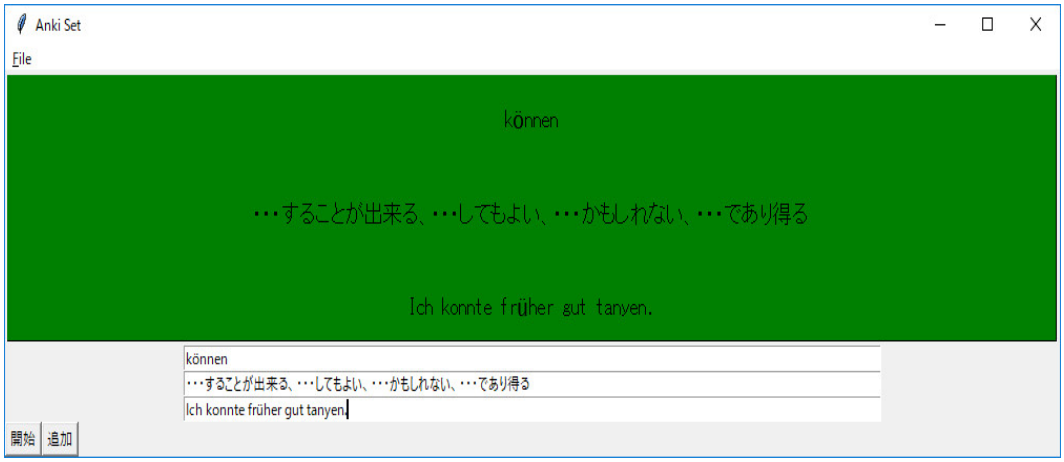
```
        for line in f:
            line = line[:-1]
            if line == '['  or line == ']': continue
            j_list = line.split(":")
            w = j_list[0][1:]
            v = j_list[1][:-1]
            words_dict[w] = v
        f.close()
    M_STOP_FLAG = False
    print('words_dict=', words_dict)
```

の書き換えとコード変換の辞書の作成が必要になります。

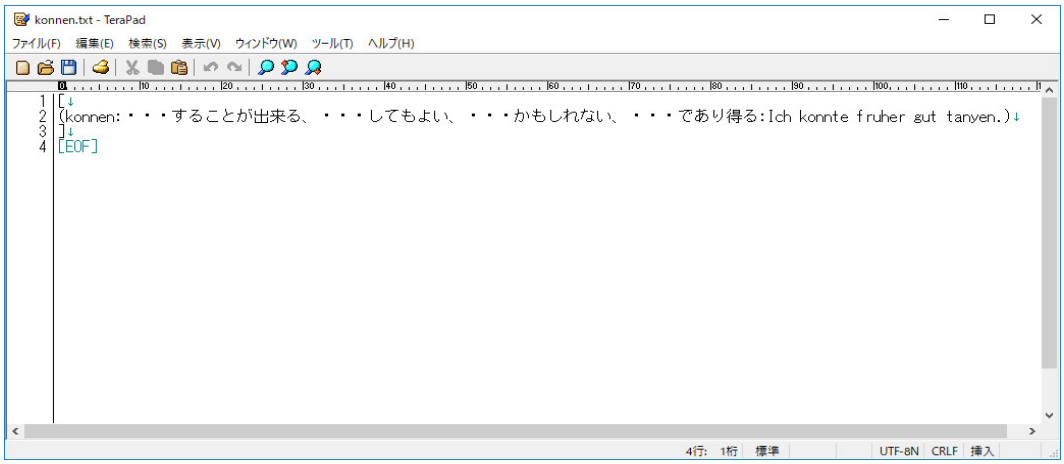
```
    f = open(filename, "w", encoding='utf-8')
```

```
    f = open(filename, "r", encoding='utf-8')
```

を使った上のプログラムでは



と入力し、保存すると



ですが、ちゃんと

のように表示してくれます。

このことは、フランス語やアラビア語を使いたいときにも同じことが起こります。

file2joseki() や teban といった変な名前がついているのは、このホームページの私の pdf ファイル「囲碁の定石を覚えるためのソフト」のプログラムからコピー・ペーストしたためで、大した意味はありません。昔作ったプログラムを修正しながらプログラミングすると、何も考えなくてもよく、素早くプログラミング出来ます。もっともゼロから考えて作った方がプログラミング技術は向上するでしょうが、プロではないので楽な道を歩んでいます。日曜大工のように、こんなものがあったらいいなと思うものを片っ端から作っていれば、少しづつプログラミング技術が向上します。Python は参考書も沢山出版されるようになりましたし、インターネットの情報も沢山あります。Python は C++ などと違って、手軽にプログラミングできる言語です。Python で試しにいろいろ作ってみて、これはというものが出来れば、C++ などに翻訳して、インターネットにアップして、誰でも無料で使えるようにすればいいです。「ものを作ること」は楽しいことです。「知ること」は楽しいことです。「知らないということ」は悲しいことです。